# What Matters in Learning from Offline Human Demonstrations for Robot Manipulation

**Ajay Mandlekar**[1], **Danfei Xu**[1], **Josiah Wong**[1], **Soroush Nasiriany**[2], **Chen Wang**[1],

**Rohun Kulkarni**[1], **Li Fei-Fei**[1], **Silvio Savarese**[1], **Yuke Zhu**[2], **Roberto Martín-Martín**[1]

[1]Stanford University, [2]The University of Texas at Austin

## Abstract

Imitating human demonstrations is a promising approach to endow robots with various manipulation capabilities. While recent advances have been made in imitation learning and batch (offline) reinforcement learning, a lack of open-source human datasets and reproducible learning methods make assessing the state of the field difficult. In this paper, we conduct an extensive study of six offline learning algorithms for robot manipulation on five simulated and three real-world multi-stage manipulation tasks of varying complexity, and with datasets of varying quality. Our study analyzes the most critical challenges when learning from offline human data for manipulation. Based on the study, we derive a series of lessons including the sensitivity to different algorithmic design choices, the dependence on the quality of the demonstrations, and the variability based on the stopping criteria due to the different objectives in training and evaluation. We also highlight opportunities for learning from human datasets, such as the ability to learn proficient policies on challenging, multi-stage tasks beyond the scope of current reinforcement learning methods, and the ability to easily scale to natural, real-world manipulation scenarios where only raw sensory signals are available. We have open-sourced our datasets and all algorithm implementations to facilitate future research and fair comparisons in learning from human demonstration data at https://arise-initiative.github.io/robomimic-web/

## 1 Introduction

Human supervision has been at the heart of the most significant recent advances in several domains such as computer vision [1–4] and natural language processing [5–7]. By intelligently extracting information from large-scale human-labeled datasets, autonomous machines have been able to reach near- or even super-human performance on decades-old problems such as image recognition and question answering. Roboticists have also attempted to tackle robot manipulation through learning from human datasets, using the paradigms of Imitation Learning [8–10] and Batch (Offline) Reinforcement Learning [11–13], where datasets consisting of robot arm trajectories, action labels at each timestep, and possibly reward labels, are used to train closed-loop policies.

As in other domains, large offline datasets offer several benefits such as scale, portability, and reproducible evaluations to measure progress. Recently, there has been considerable progress in offline learning for robot manipulation from human demonstrations [9, 14, 10]. Despite these advances, the offline learning paradigm has not been nearly as disruptive in robotics as in other disciplines – there is a large gap between autonomous robot manipulation capabilities and the wide range of tasks that humans can solve effortlessly using physical and cognitive intelligence. What has inhibited the use of large human-provided datasets to address this gap?

(a) Lift      (b) Can      (c) Tool Hang      (d) Square

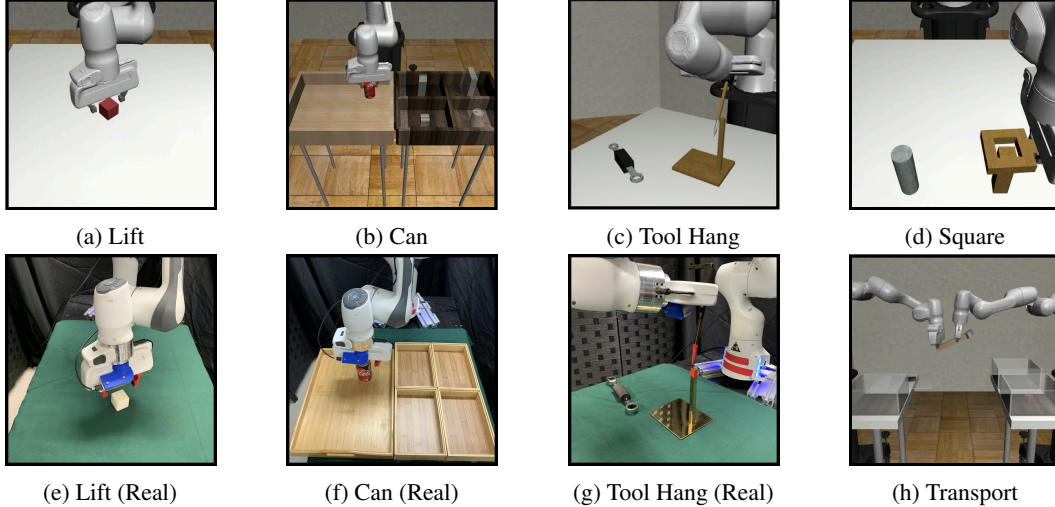(e) Lift (Real)      (f) Can (Real)      (g) Tool Hang (Real)      (h) Transport

Figure 1: **Tasks.** We collect datasets across 6 operators of varying proficiency and evaluate offline policy learning methods on 8 challenging manipulation tasks that test a wide range of manipulation capabilities including pick-and-place, multi-arm coordination, and high-precision insertion and assembly.

In contrast to other domains where supervised learning has been successful, robotic manipulation is a time-evolving dynamical system, requiring fine-grained real-time control to guide robot arms successfully through tasks – consequently, data collection can present technical challenges requiring specialized systems [15], which can explain why large-scale human-provided datasets [16, 17] have not been very prevalent. Learning from such datasets can also present several challenges. Human demonstrations can differ from machine-generated datasets (a recent trend in benchmarks for offline policy learning [18, 19]) due to a non-Markovian decision process, since humans may not act purely based on the current observation. There can also be significant variance in both data quality and solution strategy when collecting data from multiple humans [20]. Differences from classic supervised learning, such as a mismatch between training and evaluation objectives (task success rate), can make selecting a final policy challenging [21, 22], especially in real-world settings where evaluating each policy on a robot can be infeasible. Finally, offline learning is sensitive to state and action space coverage (dataset size) and agent design decisions.

Studying these challenges in the context of robot manipulation and human-provided datasets could be a stepping stone to closing the gap between robot and human manipulation capabilities. Unfortunately, a lack of suitable benchmark and human datasets have made studying this setting difficult. Prior works are either limited to studying simple 2D environments [23] or using data generated from hard-coded policies [24, 25]. In this paper, we address this need by presenting a study of data-driven offline policy learning methods on several human-provided robot manipulation datasets. We collect task demonstrations from human teleoperators across a broad range of simulated and real world manipulation tasks and investigate several factors that play a role in learning from such data.

From our results, we point out several lessons to guide future research in leveraging human supervision for robot manipulation effectively. We find that history-dependent models can be extremely effective in learning from single and multi-human datasets while state-of-the-art batch RL algorithms struggle to learn from such datasets, and that the choice of observation space and hyperparameters play a substantial role in training proficient policies. We also find that there is substantial promise for solving more complex tasks using large-scale human datasets and that our insights directly transfer to real-world scenarios, making this an important setting to explore further.

## 2    Challenges in Offline Learning from Human Datasets

In this section, we outline five challenges in offline learning from human datasets that motivate different factors that we investigate in our study.

**(C1) Data from Non-Markovian Decision Process.** Human demonstrations can differ substantially from machine-generated demonstrations because humans may not act purely based on a single current observation. External factors (teleoperation device, past actions, history of episode) may all play a

| Dataset | BC | BC-RNN | BCQ | CQL | HBC | IRIS |
|---|---|---|---|---|---|---|
| Lift (MG) | $65.3 \pm 2.5$ | $70.7 \pm 3.4$ | $\mathbf{91.3 \pm 1.9}$ | $64.0 \pm 2.8$ | $47.3 \pm 4.1$ | $96.0 \pm 1.6$ |
| Can (MG) | $64.7 \pm 3.4$ | $68.7 \pm 2.5$ | $\mathbf{75.3 \pm 0.9}$ | $1.3 \pm 0.9$ | $40.7 \pm 3.4$ | $48.0 \pm 6.5$ |
| Lift (PH) | $\mathbf{100.0 \pm 0.0}$ | $\mathbf{100.0 \pm 0.0}$ | $100.0 \pm 0.0$ | $92.7 \pm 5.0$ | $\mathbf{100.0 \pm 0.0}$ | $100.0 \pm 0.0$ |
| Can (PH) | $95.3 \pm 0.9$ | $\mathbf{100.0 \pm 0.0}$ | $88.7 \pm 0.9$ | $38.0 \pm 7.5$ | $\mathbf{100.0 \pm 0.0}$ | $100.0 \pm 0.0$ |
| Square (PH) | $78.7 \pm 1.9$ | $\mathbf{84.0 \pm 0.0}$ | $50.0 \pm 4.9$ | $5.3 \pm 2.5$ | $82.6 \pm 0.9$ | $78.7 \pm 2.5$ |
| Transport (PH) | $17.3 \pm 2.5$ | $\mathbf{71.3 \pm 6.6}$ | $7.3 \pm 3.3$ | $0.0 \pm 0.0$ | $48.6 \pm 3.8$ | $41.3 \pm 3.4$ |
| Tool Hang (PH) | $\mathbf{29.3 \pm 0.9}$ | $19.3 \pm 5.0$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $30.0 \pm 7.1$ | $11.3 \pm 2.5$ |
| Lift (MH) | $\mathbf{100.0 \pm 0.0}$ | $\mathbf{100.0 \pm 0.0}$ | $100.0 \pm 0.0$ | $56.7 \pm 40.3$ | $\mathbf{100.0 \pm 0.0}$ | $100.0 \pm 0.0$ |
| Can (MH) | $86.0 \pm 4.3$ | $\mathbf{100.0 \pm 0.0}$ | $62.7 \pm 8.2$ | $22.0 \pm 5.7$ | $91.3 \pm 2.5$ | $92.7 \pm 0.9$ |
| Square (MH) | $52.7 \pm 6.6$ | $\mathbf{78.0 \pm 4.3}$ | $14.0 \pm 4.3$ | $0.7 \pm 0.9$ | $60.7 \pm 5.0$ | $52.7 \pm 5.0$ |
| Transport (MH) | $11.3 \pm 2.5$ | $\mathbf{65.3 \pm 7.4}$ | $2.6 \pm 0.9$ | $0.0 \pm 0.0$ | $14.0 \pm 1.6$ | $10.7 \pm 0.9$ |

Table 1: **Results on Low-Dimensional Observations.** We present success rates averaged over 3 seeds for each method across the low-dim Machine-Generated (MG), Proficient-Human (PH), and Multi-Human (MH) datasets. The results show that methods that model temporal correlations (BC-RNN, HBC, IRIS) exhibit strong performance on human datasets. Furthermore, while Batch RL algorithms like BCQ are proficient on machine-generated data, they perform poorly on human datasets.

role. Prior work [20] has noted substantial benefits from leveraging models that are history-dependent and / or with temporal abstraction to learn from human demonstrations. We investigate various design choices related to such architectures in this study.

**(C2) Variance in Demonstration Quality from Multiple Humans.** Prior work [20, 17] has found that data collected from several humans can differ substantially in both demonstration proficiency and solution strategy. Differences in supervisor proficiency can manifest in many ways, such as large variations in trajectory length and noise in robot movement or mistakes (e.g. missed grasps). In our study, we evaluate offline policy learning algorithms on such datasets. While recent batch RL algorithms have shown an excellent ability to learn from mixed quality machine-generated datasets [26, 27], we empirically find that they fail to learn well from mixed quality human data.

**(C3) Dependence on Dataset Size.** Offline policy learning is sensitive to the state and action space coverage in the dataset, and by extension, the size of the dataset itself. In our study, we investigate how dataset sizes affect policy performance. This analysis is useful to understand the value of adding more data – an important consideration since collecting human demonstrations can be costly.

**(C4) Mismatch between Training and Evaluation Objectives.** Unlike traditional supervised learning, where model selection can be achieved by using the model with the lowest validation loss [21], offline policy learning often suffers from the fact that the training objective is only a surrogate for the true objective of interest (e.g. task success rate), and policy performance can change significantly from epoch to epoch. This makes it difficult to select the best trained model [19, 28, 29]. In our study, we evaluate each policy checkpoint online in the environment in simulation, and report the best policy success rate per training run. We use these ground-truth values to understand the effectiveness of different selection criteria, and confirm that offline policy selection is an important problem, especially in real-world scenarios where large-scale empirical evaluation is difficult.

**(C5) High Sensitivity to Agent Design Decisions.** Prior studies on machine-generated datasets have shown that offline policy learning can be extremely sensitive to hyperparameter choices [19, 28]. In our study, we explore how agent design decisions affect policy performances, including the choice of agent architecture, agent observation space, and hyperparameter choices per algorithm. This results in several practical conclusions that should prove useful to researchers and practitioners alike. We further show that important design decisions made through our study in simulation directly translate to effective policy learning on real world tasks and datasets.

## 3 Study Design

### 3.1 Tasks

We conducted our study across 5 simulated and 3 real world tasks. The tasks were chosen to test a broad range of manipulation capabilities. See Fig 1 and Appendix E for more details.

**Lift (sim + real)).** The robot arm must lift a small cube. This is the simplest task.

| Dataset | BC | BC-RNN | BCQ | CQL | HBC | IRIS |
|---|---|---|---|---|---|---|
| Can-Worse | 56.7±2.5 | **92.0±1.6** | 29.3±10.9 | 4.0±3.3 | 78.7±3.4 | 77.3±1.9 |
| Can-Okay | 72.0±2.8 | **95.3±1.9** | 58.0±8.6 | 22.0±4.3 | 97.3±0.9 | 96.0±0.0 |
| Can-Better | 83.3±2.5 | **99.3±0.9** | 62.0±5.9 | 20.7±7.4 | 96.7±0.9 | 96.0±0.0 |
| Can-Worse-Okay | 74.7±5.7 | **98.7±1.9** | 50.7±3.8 | 18.7±2.5 | 88.0±1.6 | 87.3±1.9 |
| Can-Worse-Better | 76.0±4.3 | **100.0±0.0** | 48.0±4.9 | 20.7±5.7 | 90.0±1.6 | 91.3±2.5 |
| Can-Okay-Better | 90.7±1.9 | **100.0±0.0** | 68.7±2.5 | 30.7±7.7 | 99.3±0.9 | 98.0±1.6 |
| Square-Worse | 22.0±4.3 | 39.3±3.8 | 5.3±1.9 | 0.0±0.0 | **44.7±6.8** | 38.7±0.9 |
| Square-Okay | 27.3±3.4 | 45.3±2.5 | 6.7±1.9 | 0.0±0.0 | **52.0±2.8** | 42.0±3.3 |
| Square-Better | 58.7±2.5 | **66.0±2.8** | 32.0±4.3 | 0.7±0.9 | 61.3±1.9 | 60.0±1.6 |
| Square-Worse-Okay | 28.7±2.5 | **55.3±0.9** | 8.7±1.9 | 2.7±1.9 | 50.7±4.1 | 43.3±2.5 |
| Square-Worse-Better | 46.7±5.7 | **73.3±6.2** | 15.3±2.5 | 1.3±0.9 | 65.3±3.4 | 56.7±3.4 |
| Square-Okay-Better | 56.7±4.1 | **74.0±2.8** | 22.0±4.3 | 1.3±0.9 | 63.3±4.1 | 56.7±3.8 |
| Can-Paired | 64.0±9.1 | 70.0±4.3 | 44.7±1.9 | 6.0±1.6 | 70.7±5.2 | **75.3±1.9** |

Table 2: **Results on Suboptimal Human Data.** We present success rates averaged over 3 seeds for each method across different subsets of the Multi-Human datasets, corresponding to mixtures of demonstrations from "Better", "Adequate", and "Worse" human operators, and finally on a diagnostic dataset with paired success and failure human trajectories for each starting initialization. Results indicate that BC-RNN is a strong baseline, and that Batch RL methods perform poorly across all datasets, even on the simple diagnostic dataset.

**Can (sim + real).** The robot must place a coke can from a large bin into a smaller target bin. Slightly more challenging than Lift, since picking the can is harder than picking the cube, and the can must also be placed into the bin.

**Square (sim).** The robot must pick a square nut and place it on a rod. Substantially more difficult than Lift and Pick Place Can due to the precision needed to pick up the nut and insert it on the rod.

**Transport (sim).** Two robot arms must transfer a hammer from a closed container on a shelf to a target bin on another shelf. One robot arm must retrieve the hammer from the container, while the other arm must clear the target bin by moving a piece of trash to the nearby receptacle. Finally, one arm must hand the hammer over to the other, which must place the hammer in the target bin.

**Tool Hang (sim + real).** A robot arm must assemble a frame consisting of a base piece and hook piece by inserting the hook into the base, and hang a wrench on the hook. This is the most difficult task due to the multiple stages that each require precise, and dexterous, rotation-heavy movements.

## 3.2 Data Collection

To study the effect of dataset source, we collected data from three different sources – Machine-Generated, Proficient-Human, and Multi-Human (more details in Appendix B).

**Machine-Generated (MG).** We collected these datasets by first training a state-of-the-art RL algorithm [30] on the Lift and Can task, taking agent checkpoints that are saved regularly during training, and collecting 300 rollout trajectories from each checkpoint. Consequently, these datasets are comprised of mixtures of expert and suboptimal data, and resemble datasets from common offline RL benchmarks [18, 19]. We excluded other tasks because they could not be solved by the RL algorithm even with substantial tuning. See the appendix for more details.

**Proficient-Human (PH) and Multi-Human (MH).** Datasets are collected by humans through RoboTurk [15, 17], a remote teleoperation platform. The PH datasets consist of 200 demonstrations collected by a single, experienced teleoperator, while the MH datasets consist of 300 demonstrations, collected by 6 teleoperators of varying proficiency, each of which provided 50 demonstrations. The 6 teleoperators consisted of a "better" group of 2 experienced operators, an "okay" group of 2 adequate operators, and a "worse" group of 2 inexperienced operators. These data subsets in the Multi-Human data allowed us to investigate the ability of algorithms to deal with mixed quality human data.

**Observation Modalities.** To study the effect of observation modalities, we capture a diverse set of sensor streams when collecting the dataset, including end-effector, gripper fingers, and joints, ground-truth object poses, and images from an external camera and wrist-mounted camera per robot arm (see Appendix E). We have two observation spaces – "low-dim" and "image". Both include end-effector poses and gripper finger positions, and only differ in whether ground-truth object information is used (low-dim) or whether that information is replaced by the available camera observations (image).

(a) Observation Space     (b) Low-Dim Hyperparameters     (c) Image Hyperparameters
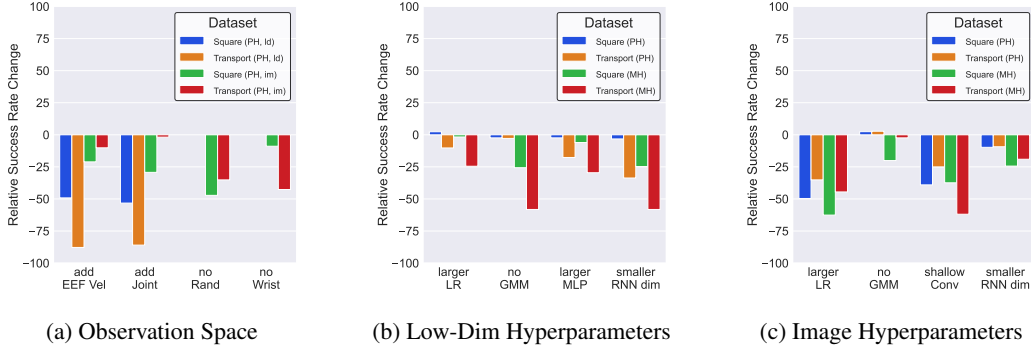
Figure 2: **Effect of Observation Space and Hyperparameter Choice.** We show how the success rate that BC-RNN obtains can drop drastically due to changes to the observation space and hyperparameter settings.
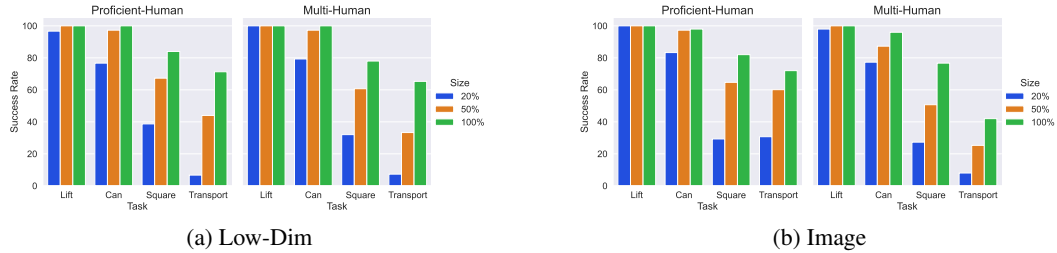


(a) Low-Dim            (b) Image

Figure 3: **Effect of Dataset Size.** We study how the BC-RNN success rate changes when lowering the quantity of data to 20% and 50%. Results show that less complex tasks (Lift, Can) be learned with a fraction of the data, while more complex tasks might benefit from even larger human datasets.

## 3.3 Training and Evaluation Protocols

There are several approaches to offline imitation learning [31–34, 9, 10, 25, 35, 36] and offline reinforcement learning [26, 27, 37–43] (see Appendix A for more discussion on related work). We chose to evaluate 6 algorithms in this study – Behavioral Cloning (BC), BC with an RNN policy (BC-RNN), Hierarchical Behavioral Cloning (HBC) [10], Batch-Constrained Q-Learning (BCQ) [26], Conservative Q-Learning (CQL) [27], and IRIS [20]. BC-RNN, HBC, and IRIS have all been used in prior work to learn offline from teleoperated human demonstrations, while BCQ and CQL are commonly-used offline RL algorithms (see Appendix C). We use binary task completion rewards for all our experiments. Each agent is trained for $N$ epochs, where each epoch consists of $M$ gradient steps, and evaluated every $E$ epochs, by running 50 rollouts in the environment and reporting the success rate over a maximum horizon. For each agent, we report the maximum success rate over the coarse of training, and average over 3 seeds. For low-dim agents, $N = 2000$, $M = 100$, and $E = 50$, and for image agents, $N = 600$, $M = 500$, and $E = 20$ (see Appendix B.2).

## 4 Experiments

In this section, we present each factor that we explored in our study, and note the relevant challenges from Sec. 2 that each pertains to.

### 4.1 Algorithm Comparison on Single and Multi-Human Demonstrations (C1, C2)

We trained and evaluated all algorithms on the Proficient-Human (PH) and Multi-Human (MH) datasets and report the average success rates across 3 seeds in Table 1.

**Observation history is crucial for good performance.** There is a substantial performance gap between BC-RNN and BC, which highlights the benefits of history-dependence. The performance gap is larger for longer-horizon tasks (e.g. $\sim 55\%$ for Transport (PH) compared to $\sim 5\%$ for Square (PH)) and for multi-human data compared to single-human data (e.g. $\sim 25\%$ for Square (MH) compared to $\sim 5\%$ for Square (PH)). Interestingly, results are lower for MH datasets compared to PH datasets,

even though the MH datasets contain 100 more demos (300 demos vs. 200 demos). This most likely stems from the presence of suboptimal and multimodal data in the MH datasets.

**Batch RL algorithms perform poorly on Human Datasets.** Recent batch (offline) RL algorithms such as BCQ and CQL have demonstrated excellent results in learning from suboptimal and multi-modal agent-generated datasets. Our results confirm the capacity of such algorithms to work well – BCQ in particular performs strongly on our agent-generated MG datasets that consist of a diverse mixture of good and poor policies. Surprisingly though, neither BCQ nor CQL performs particularly well on these human-generated datasets. This puts the ability of such algorithms to learn from more natural dataset distributions into question (instead of those collected via RL exploration or pre-trained agents). There is an opportunity for future work in batch RL to resolve this gap.

## 4.2 Learning from Suboptimal Human Data (C2)

To further investigate how algorithms deal with suboptimal human data, we split our MH datasets into smaller subsets based on the proficiency of the human operators. The MH-Better, MH-Okay, and MH-Worse are the 100 demo subsets corresponding to the 2 "better", 2 "okay", and 2 "worse" operators respectively, while MH Worse-Okay, MH Worse-Better, and MH Okay-Better are the 200 demo subsets corresponding to the mixture of the previous subsets. Similar data mixtures have been used for evaluations in batch RL [18]. Appendix B shows the average trajectory lengths in each data subset – lower quality datasets contain demonstrations that take more time to solve the task.

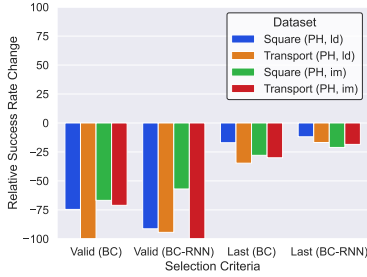**BC-RNN is a strong baseline on suboptimal human data, but there is room for improvement.** Table 2 shows that BC exhibits a large performance gap between the Better and Worse 100-demo subsets (roughly 27% and 35% for Can and Square respectively). Interestingly, BC-RNN is able to nearly eliminate this gap in performance on the Can task, but not on the Square task. However, BC-RNN outperforms BC on all datasets (7%-35% improvement). Comparing results on the 100 Better demonstrations and 100 Okay demonstrations to the 200 Worse-Better demonstrations and 200 Worse-Okay demonstrations further allows us to analyze how adding 100 "worse" demonstrations impacts the performance of each algorithm. Most algorithms decline in performance while BC-RNN is able to uniformly improve from the added data. Comparing the performance of BC-RNN on the 200-demo Square mixture datasets (55.3%, 73.3%, 74.0%) to the high-quality 200-demo Square (PH) dataset (84.0%) shows that there is still room for algorithms to improve on the use of this data.

**Diagnostic dataset shows that Batch RL struggles in simpler settings as well.** The final row of Table 2 shows additional results on a diagnostic dataset termed Can-Paired, where a single operator collected 2 demonstrations for each of 100 task initializations – one successful demonstration, and one where the can is tossed outside of the bin (task failure), for a total of 200 demonstrations. There is a strong expectation for batch RL algorithms to be able to distinguish between actions leading to successful placement and actions leading to task failure, but even in this simple setting, most algorithms suffer, providing a pessimistic view of the state-of-the-art. The 5% improvement that IRIS provides over BC-RNN suggests that introducing history-dependence into state-of-the-art batch RL algorithms might be a promising direction for future work.

## 4.3 Effect of Observation Space (C5)

**Learning from image observations can match low-dim agent performance.** In Table 3, we present policy learning results when using image observations instead of ground-truth object locations – an important setting for real-world policy learning. BC-RNN still maintains superior performance improvements over BC on the complex Square and Transport tasks, and with the exception of Transport (MH), maintains nearly the same performance as learning from ground-truth observations. This result provides an optimistic view for learning with real-world raw sensory observations.

**Features used for robot proprioception can matter.** In Fig 2a, we study the effect of adding end effector velocities to the observations (+ EEF Vel), and joint positions and velocities to the observations (+ Joint). Surprisingly, we find that including end effector velocity information, and joint information hurts agents trained on low-dim observations substantially (49%-88% relative performance drop), while image-based agents are more tolerant to the inclusion of this extra information (2%-29% relative performance drop). We hypothesize that performance drops might be due to overfitting to the presence of this extra information not needed for solving these tasks. Thus, practitioners

(a) **Effect of Policy Selection Criteria**

Table 3: **Results on Image Observations.**

| Dataset | BC | BC-RNN | BCQ | CQL |
|---|---|---|---|---|
| Lift (PH) | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $98.0 \pm 1.6$ | $52.0 \pm 13.0$ |
| Can (PH) | $97.3 \pm 1.9$ | $98.0 \pm 0.9$ | $86.7 \pm 2.5$ | $0.7 \pm 0.9$ |
| Square (PH) | $62.0 \pm 4.9$ | $82.0 \pm 0.0$ | $41.3 \pm 4.1$ | - |
| Transport (PH) | $55.3 \pm 6.2$ | $72.0 \pm 4.3$ | $0.7 \pm 0.9$ | - |
| Tool Hang (PH) | $20.0 \pm 5.9$ | $67.3 \pm 4.1$ | $3.3 \pm 0.9$ | - |
| Lift (MH) | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $93.3 \pm 0.9$ | $11.3 \pm 9.3$ |
| Can (MH) | $85.3 \pm 0.9$ | $96.0 \pm 1.6$ | $77.3 \pm 6.8$ | $0.0 \pm 0.0$ |
| Square (MH) | $46.0 \pm 1.6$ | $76.7 \pm 3.4$ | $17.3 \pm 7.5$ | - |
| Transport (MH) | $18.7 \pm 2.5$ | $42.0 \pm 1.6$ | $0.0 \pm 0.0$ | - |

Figure 5: (left) **Effect of Policy Selection Criteria.** We compare how performance decreases when choosing the policy to evaluate by using the lowest validation loss, or when using the final trained checkpoint, with respect to the best policy performance. (right) **Results on Image Observations.** We present success rates for each method across the image observation human datasets. BC-RNN maintains nearly the same performance as learning from ground-truth observations, providing an optimistic view for learning with real-world raw sensory observations.

should take care to engineer the robot observation space and exclude possibly irrelevant information – information-hiding can be a powerful paradigm for training proficient robots [44].

**Image randomization and wrist observations can be crucial for manipulation tasks.** In Fig 2a, we report performance drops from removing pixel shift image randomization (- Rand) and the wrist camera (- Wrist) from image-based agents to understand their importance. We see that removing randomization results in 47% and 35% relative performance drops on Square and Transport respectively, and removing wrist images results in 9% and 43% relative drops. Consequently, both wrist camera images and image randomization play a substantial role in producing performant policies. We confirm the importance of each for visuomotor imitation in the real world as well (see Sec 4.7). Wrist observations likely help the robot improve gripper alignment for grasping and randomization helps the policy develop invariance for portions of the image that are not important for action prediction.

## 4.4 Effect of Hyperparameter Choice (C5)

In this section, we take our default hyperparameters for BC-RNN and study the effect of changing a subset of them to report practical recommendations for learning from human datasets (see Appendix I for BCQ and CQL). We present our results in Fig 2b (low-dim) and Fig 2c (image).

**(larger LR)** Increasing the learning rate from 1e-4 to 1e-3 affects the performance of image-agents substantially (drop of 35%-63%), while low-dim agents are more tolerant to the change. **(no GMM)** Using a deterministic policy instead of learning a GMM action distribution results in significant relative performance drops on the MH datasets (especially low-dim Transport, with a drop of 58%). **(larger MLP)** Using a larger MLP size at each RNN timestep reduces performance uniformly, suggesting that it is possible to overfit to dataset actions if network architectures are too large. **(shallow Conv)** Using a shallow convolutional network [45] instead of the ResNet backbone [46] for encoding image observations reduces performance significantly – with relative drops of 25%-62%, suggesting that large-capacity visual encoders are crucial for visuomotor imitation. **(smaller RNN dim)** Reducing the size of the RNN hidden dimension from 400 to 100 (low-dim) and 1000 to 400 (image) uniformly decreases performance (drops of 3%-58%), showing the importance of a large RNN hidden dimension. **(Recommendations)** We recommend tuning the LR (especially for image agents) and network structure (MLP size, size of RNN dim) carefully. Opting to use a GMM policy and a ResNet encoder appears to be uniformly better.

## 4.5 Selecting a Policy to Evaluate (C4)

Model selection in offline policy learning can be challenging – for this reason, in our simulation experiments, we evaluated every policy checkpoint online and reported the best one. This is not feasible for real-world settings, making offline policy selection desirable. In Fig 4a, we show that this can be non-trivial, by showing the relative performance drop when selecting the policy using the best loss on validation data (common in supervised learning), and when using the final training checkpoint as well (common in offline RL [47, 18, 27]) – in both cases, the selected policy is significantly worse

than the best one (10% to 100% decrease). See Appendix G for more detailed results and discussion. This motivates the need for better offline evaluation metrics.

## 4.6 Effect of Dataset Size (C3)

To study how dataset size impacts performance, we formed smaller 20% and 50% subsets of our human datasets by sampling trajectories. We evaluate low-dim and image BC-RNN agents across these subsets in Table 27 and Table 28. There are several promising results here. We first note that less complex tasks (Lift, Can) can yield proficient policies (75%-100% success rate) using a small fraction of the data (20%). Second, while policies trained on more complex tasks (Square, Transport) suffer substantially when using 50% or 20% of the data, the converse is also true – adding more data (e.g. moving from 20% to 50% or 50% to 100% size) can result in significant policy improvement. This confirms the value of using large human datasets as a means to obtain proficient policies for challenging and complex manipulation tasks.

## 4.7 Applicability to Real-World Settings

Here, we show that design decisions made in simulation can potentially transfer to real world settings. We collected 3 additional real-world datasets with a Franka robotic arm – Lift (Real), Can (Real), and Tool Hang (Real). Each consists of 200 trajectories collected by one operator. We train BC-RNN and report the final policy checkpoint success rate, over 30 rollouts, due to the time-consuming nature of real world policy evaluation. We also emphasize that no real-world hyperparameter tuning took place, so our results are a lower bound. We were able to train proficient **Lift (96.7%)** and **Can (73.3%)** policies, and the **Tool Hang (3.3%)** policy is able to generate some task successes, despite the extremely difficult nature of the task. Furthermore, as in Sec. 4.3, we validate the importance of pixel shift randomization and the wrist camera by ablating each component on the Can task, and show that including both is the difference between a proficient and non-proficient real-world policy – **Can (- Rand) (26.7%)**, **Can (- Wrist) (43.3%)**.

## 5 Discussion

In this section, we summarize the lessons from our study and make recommendations for future work.

**(L1) Models with temporal abstraction can be extremely effective in learning from human datasets.** In Sec 4.1 and Sec 4.2, we demonstrated that history-dependent models (BC-RNN, HBC, and IRIS) are particularly effective in learning from human datasets compared to algorithms that do not take temporal context into account.

**(L2) Need to improve the ability of batch (offline) RL to learn from suboptimal human datasets.** Sec 4.2 and Appendix I demonstrated that state-of-the-art batch RL algorithms are excellent at learning from suboptimal machine-generated datasets but much worse at learning from suboptimal human datasets. They even struggled with a diagnostic dataset with paired good and bad human demonstration trajectories while IRIS was able to improve slightly on BC-RNN, suggesting that combining history-dependence with value learning might be a good place to start for improving batch RL methods [48–51]. This also demonstrates a need to start benchmarking new batch RL algorithms on human datasets instead of purely on machine-generated datasets.

**(L3) Improving offline policy selection is important for real world settings.** Sec 4.5 demonstrated the need for better ways to select an evaluation policy in an offline manner. We hope that our datasets can help supplement other efforts [29].

**(L4) Observation space plays a large role and hyperparameters matter.** Sec 4.3 demonstrates that policies trained on low-dim observations can be very sensitive to the choice of robot proprioception, while pixel shift randomization and wrist camera images are critical for effective visuomotor policy learning. The choice of observation space for imitation merits careful consideration – other work has also confirmed the importance of feature representations used for offline policy learning [52, 53]. Sec 4.4 and Appendix I made practical recommendations for choosing hyperparameters to learn from human data.

**(L5) There is substantial promise for solving more complex tasks using large-scale human datasets.** Sec 4.6 showed that adding more data can result in significant policy improvement on

complex tasks. Table 3 and Sec 4.7 shows that we could learn proficient policies on the Tool Hang task, our most complex task, without any hyperparameter tuning on the task or dataset. Together, these results show the potential of large human datasets as a means to solve challenging and complex manipulation tasks.

**(L6) Study results transfer to real-world settings.** In Sec 4.7, we showed that we could directly apply hyperparameters that were tuned on simulated tasks directly to real-world datasets and tasks. This provides promise for using our tasks, datasets, and codebase to enable reproducible evaluation in simulation, while also being confident that conclusions can transfer to real-world settings.

Going forward, we hope that the datasets, tasks, code, and subsequent insights of our study will serve researchers and practitioners alike.

## Acknowledgments and Disclosure of Funding

## References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[5] P. Rajpurkar, R. Jia, and P. Liang, "Know What You Don't Know: Unanswerable Questions for SQuAD," *arXiv preprint arXiv:1806.03822*, 2018.

[6] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, no. 4, pp. 681–694, 2020.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[8] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, pp. 305–313, 1989.

[9] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," *arXiv preprint arXiv:1710.04615*, 2017.

[10] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020.

[11] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*, pp. 45–73, Springer, 2012.

[12] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[13] S. Cabi, S. Gómez Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *arXiv*, pp. arXiv–1909, 2019.

[14] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 492–499, 2019.

[15] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei, "RoboTurk: A Crowdsourcing Platform for Robotic Skill Learning through Imitation," in *Conference on Robot Learning*, 2018.

[16] P. Sharma, L. Mohan, L. Pinto, and A. Gupta, "Multiple interactions made easy (mime): Large scale demonstrations data for imitation," in *Conference on robot learning*, pp. 906–915, PMLR, 2018.

[17] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity," *arXiv preprint arXiv:1911.04052*, 2019.

[18] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[19] C. Gulcehre, Z. Wang, A. Novikov, T. L. Paine, S. G. Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. Mankowitz, C. Paduraru, *et al.*, "Rl unplugged: Benchmarks for offline reinforcement learning," *arXiv preprint arXiv:2006.13888*, 2020.

[20] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420, IEEE, 2020.

[21] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[22] D. Precup, "Eligibility traces for off-policy policy evaluation," *Computer Science Department Faculty Publication Series*, p. 80, 2000.

[23] S. Toyer, R. Shah, A. Critch, and S. Russell, "The magical benchmark for robust imitation," *arXiv preprint arXiv:2011.00401*, 2020.

[24] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[25] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," *arXiv preprint arXiv:2010.14406*, 2020.

[26] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*, pp. 2052–2062, PMLR, 2019.

[27] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *arXiv preprint arXiv:2006.04779*, 2020.

[28] T. L. Paine, C. Paduraru, A. Michi, C. Gulcehre, K. Zolna, A. Novikov, Z. Wang, and N. de Freitas, "Hyperparameter selection for offline reinforcement learning," *arXiv preprint arXiv:2007.09055*, 2020.

[29] J. Fu, M. Norouzi, O. Nachum, G. Tucker, Z. Wang, A. Novikov, M. Yang, M. R. Zhang, Y. Chen, A. Kumar, *et al.*, "Benchmarks for deep off-policy evaluation," *arXiv preprint arXiv:2103.16596*, 2021.

[30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, pp. 1861–1870, PMLR, 2018.

[31] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," *Proceedings 2002 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1398–1403 vol.2, 2002.

[32] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in *Conference in Robot Learning*, vol. abs/1709.04905, 2017.

[33] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, 2008.

[34] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 44–54, 2010.

[35] C. Wang, R. Wang, D. Xu, A. Mandlekar, L. Fei-Fei, and S. Savarese, "Generalization through hand-eye coordination: An action space for learning spatially-invariant visuomotor control," *arXiv preprint arXiv:2103.00375*, 2021.

[36] A. Tung, J. Wong, A. Mandlekar, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Learning multi-arm manipulation through collaborative teleoperation," *arXiv preprint arXiv:2012.06738*, 2020.

[37] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," *arXiv preprint arXiv:1911.11361*, 2019.

[38] Z. Wang, A. Novikov, K. Żołna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, *et al.*, "Critic regularized regression," *arXiv preprint arXiv:2006.15134*, 2020.

[39] N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller, "Keep doing what worked: Behavioral modelling priors for offline reinforcement learning," *arXiv preprint arXiv:2002.08396*, 2020.

[40] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," *arXiv preprint arXiv:2005.05951*, 2020.

[41] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," *arXiv preprint arXiv:2005.13239*, 2020.

[42] S. K. S. Ghasemipour, D. Schuurmans, and S. S. Gu, "Emaq: Expected-max q-learning operator for simple yet effective offline and online rl," *arXiv preprint arXiv:2007.11091*, 2020.

[43] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," *arXiv preprint arXiv:2102.08363*, 2021.

[44] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[45] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519, IEEE, 2016.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[47] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," in *International Conference on Machine Learning*, pp. 104–114, PMLR, 2020.

[48] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "Opal: Offline primitive discovery for accelerating offline reinforcement learning," *arXiv preprint arXiv:2010.13611*, 2020.

[49] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," *arXiv preprint arXiv:2010.11944*, 2020.

[50] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *arXiv preprint arXiv:2106.01345*, 2021.

[51] M. Janner, Q. Li, and S. Levine, "Reinforcement learning as one big sequence modeling problem," *arXiv preprint arXiv:2106.02039*, 2021.

[52] M. Yang and O. Nachum, "Representation matters: Offline pretraining for sequential decision making," *arXiv preprint arXiv:2102.05815*, 2021.

[53] O. Nachum and M. Yang, "Provable representation learning for imitation with contrastive fourier features," *arXiv preprint arXiv:2105.12272*, 2021.

[54] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[55] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[56] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 391–398, 2012.

[57] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning robot objectives from physical human interaction," *Proceedings of Machine Learning Research*, vol. 78, pp. 217–226, 2017.

[58] A. Bajcsy, D. P. Losey, M. K. O'Malley, and A. D. Dragan, "Learning from physical human corrections, one feature at a time," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 141–149, 2018.

[59] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *arXiv preprint arXiv:2012.06733*, 2020.

[60] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone, "Leveraging human guidance for deep reinforcement learning tasks," *arXiv preprint arXiv:1909.09906*, 2019.

[61] R. Loftin, B. Peng, J. MacGlashan, M. L. Littman, M. E. Taylor, J. Huang, and D. L. Roberts, "Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning," *Autonomous agents and multi-agent systems*, vol. 30, no. 1, pp. 30–59, 2016.

[62] J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, D. Roberts, M. E. Taylor, and M. L. Littman, "Interactive learning from policy-dependent human feedback," *arXiv preprint arXiv:1701.06049*, 2017.

[63] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

[64] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv preprint arXiv:1904.07854*, 2019.

[65] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*, pp. 1329–1338, PMLR, 2016.

[66] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

[67] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, *et al.*, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv preprint arXiv:2006.05990*, 2020.

[68] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[69] R. Memmesheimer, I. Mykhalchyshyna, V. Seib, and D. Paulus, "Simitate: A hybrid imitation learning benchmark," *arXiv preprint arXiv:1905.06002*, 2019.

[70] L. Hussenot, M. Andrychowicz, D. Vincent, R. Dadashi, A. Raichuk, L. Stafiniak, S. Girgin, R. Marinier, N. Momchev, S. Ramos, *et al.*, "Hyperparameter selection for imitation learning," *arXiv preprint arXiv:2105.12034*, 2021.

[71] M. A. Rana, D. Chen, J. Williams, V. Chu, S. R. Ahmadzadeh, and S. Chernova, "Benchmark for skill learning from demonstration: Impact of user experience, task complexity, and start configuration on performance," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7561–7567, IEEE, 2020.

[72] A. Lemme, Y. Meirovitch, M. Khansari-Zadeh, T. Flash, A. Billard, and J. J. Steil, "Open-source benchmarking for learned reaching motion generation in robotics," *Paladyn, Journal of Behavioral Robotics*, vol. 6, no. 1, 2015.

[73] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark," in *Conference on Robot Learning*, pp. 767–782, PMLR, 2018.

[74] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," *arXiv preprint arXiv:2011.07215*, 2020.

[75] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning*, pp. 1094–1100, PMLR, 2020.

[76] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[77] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, and S. Bauer, "Causal-world: A robotic manipulation benchmark for causal structure and transfer learning," *arXiv preprint arXiv:2010.04296*, 2020.

[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[79] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[80] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[81] L. Biewald, "Experiment tracking with weights and biases," 2020. Software available from wandb.com.

[82] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

[83] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[84] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv preprint arXiv:2004.14990*, 2020.

[85] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv preprint arXiv:2004.13649*, 2020.

[86] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, "Visual imitation made easy," *arXiv e-prints*, pp. arXiv–2008, 2020.

[87] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, "A framework for efficient robotic manipulation," *arXiv preprint arXiv:2012.07975*, 2020.

[88] M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Novikov, S. G. Colmenarejo, S. Cabi, C. Gulcehre, T. L. Paine, A. Cowie, Z. Wang, B. Piot, and N. de Freitas, "Acme: A research framework for distributed reinforcement learning," *arXiv preprint arXiv:2006.00979*, 2020.

# Appendix

## A Additional Related Works

**Human Supervision for Robotics:** Prior work has tried using human supervision in the form of human demonstrations provided both by teleoperating the robot [9, 10] and via kinesthetic teaching [54–56], corrections to actions taken by the robot [21, 57–59], and positive and negative feedback for robot actions [60–62, 13, 63, 64].

**Offline Learning from Demonstrations**: Imitation Learning (IL) is a popular paradigm for training policies from a set of demonstrations. Offline Imitation Learning typically consists of variants of Behavioral Cloning (BC) [8], where a policy is trained to output the same actions as the ones taken by the demonstrator in each state. Offline IL has been used extensively in robotic manipulation [31–34, 9, 10, 25, 35, 36]. Typically, IL assumes that the demonstration data is optimal. By contrast, Batch (Offline) Reinforcement Learning [11, 12] is a method to learn from demonstrations that can consist of both good and bad quality data, by leveraging reward annotations in the datasets. Prior algorithms [26, 27, 37–43] are mostly evaluated on datasets generated by several RL-trained policies of varying quality. In this study, we evaluate both offline IL and offline RL algorithms on datasets collected from one or more humans, which can both break the assumption of optimality in IL, and present interesting challenges for offline RL methods compared to commonly used RL-generated datasets.

**Empirical Studies in Reinforcement and Imitation Learning**: Prior work has benchmarked Reinforcement Learning (RL) algorithms in continuous control domains [65], run extensive evaluations of model-based RL algorithms [66] and on-policy RL methods [67], and shown how Deep RL algorithms can be extremely sensitive to hyperparameter choices and can make reproducing results challenging [68]. There are fewer empirical studies in imitation learning. The MAGICAL benchmark [23] consists of 2D environments that test the ability of IL algorithms to generalize. Both RLBench [24] and Ravens [25] provide several simulated robot manipulation tasks and expert demonstrations for imitation learning, but demonstrators are pre-programmed and rely on ground-truth simulator state. Simitate [69] is an imitation learning benchmark suite consisting of real-world motion trajectories collected by humans, and carefully translated into simulation using extensive sensor instrumentation in the real world. By contrast, our datasets are collected via remote teleoperation from humans in both simulated and real-world settings, allowing fast and easy demonstration collection for a wide range of tasks without assuming privileged instrumentation. Recently, Hussenot et al. [70] presented an empirical study on the importance of hyperparameter selection in imitation learning – it consists of an extensive evaluation of both offline and online imitation learning algorithms using modest-sized datasets (~10s of trajectories). This work is complementary to ours – we also confirm the importance of hyperparameter selection when learning offline from human-provided datasets (~100s of trajectories).

Prior work has also established benchmarks for motion-based learning from demonstration methods [71, 72], which directly model entire robot arm trajectories. Lemme et al. [72] evaluates motion generation methods by having a robot arm reproduce several point-to-point motions from demonstrations. This differs from our focus – tabletop manipulation tasks where a robot must interact with one or more objects. Lemme et al. [72] also evaluate the robustness of the learned motions by introducing perturbations – similar mechanisms could be used to understand the robustness of the policies trained in this study. While such evaluations are important for deploying policies in real-world settings, we leave this for future work. Similar to our study, recent work by Rana et al. [71] collected demonstration trajectories across many humans and tasks. However, their datasets consist of robot end effector trajectories, while our datasets and learning methods focus on leveraging additional modalities such as object poses and camera images to train policies that can solve tasks across several scene configurations. Rana et al. [71] also used crowdsourced humans to evaluate learned robot motions with subjective metrics such as safety, while we primarily evaluate our policies using task success rate. Subjective measures like safety are important for real-world policy deployment, but this is also left for future work.

**Robot Manipulation Benchmarks**: Several benchmark robot manipulation tasks have been proposed before [73–76, 24, 77]. Benchmark datasets [18, 19] have also been proposed recently for batch (offline) reinforcement learning. While these datasets span a variety of domains (locomotion, control,

autonomous driving, video games, robot manipulation), most of the datasets are generated by autonomous agents trained with online reinforcement learning. Unfortunately, this means that most of these datasets are limited to tasks that RL methods can solve from scratch. By contrast, we focus on datasets collected by one or more humans, allowing us to increase the complexity of the tasks we consider. Furthermore, our study explores how learning from human data can be substantially different than agent datasets.

# B Dataset Details

| Dataset Type | Lift | Can | Square | Transport | Tool Hang |
|---|---|---|---|---|---|
| Machine-Generated (MG) | $150 \pm 0$ | $150 \pm 0$ | - | - | - |
| Proficient-Human (PH) | $48 \pm 6$ | $116 \pm 14$ | $151 \pm 20$ | $469 \pm 54$ | $480 \pm 88$ |
| Multi-Human (MH) | $104 \pm 44$ | $209 \pm 114$ | $269 \pm 123$ | $653 \pm 201$ | - |
| MH-Better | $72 \pm 24$ | $143 \pm 29$ | $185 \pm 46$ | $461 \pm 56$ | - |
| MH-Okay | $94 \pm 30$ | $181 \pm 47$ | $265 \pm 78$ | $636 \pm 128$ | - |
| MH-Worse | $145 \pm 40$ | $304 \pm 148$ | $357 \pm 150$ | $778 \pm 221$ | - |
| MH-Worse-Okay | $119 \pm 44$ | $242 \pm 126$ | $311 \pm 128$ | $710 \pm 115$ | - |
| MH-Worse-Better | $109 \pm 49$ | $224 \pm 134$ | $271 \pm 140$ | $734 \pm 297$ | - |
| MH-Okay-Better | $83 \pm 29$ | $162 \pm 44$ | $225 \pm 76$ | $597 \pm 83$ | - |

Table 4: **Average Trajectory Lengths by Dataset.** The table shows the average trajectory length (mean and standard deviation) for each dataset variant. This was used to determine evaluation rollout horizons for each dataset. The length is a proxy for the quality of the dataset – less proficient humans took more time to demonstrate the task.

| Dataset Type | Lift | Can | Square | Transport | Tool Hang |
|---|---|---|---|---|---|
| Machine-Generated(MG) | 400 | 400 | - | - | - |
| Proficient-Human(PH) | 400 | 400 | 400 | 700 | 700 |
| Multi-Human(MH) | 500 | 500 | 500 | 1100 | - |
| MH-Better | 500 | 500 | 500 | 1100 | - |
| MH-Okay | 500 | 500 | 500 | 1100 | - |
| MH-Worse | 500 | 500 | 500 | 1100 | - |
| MH-Worse-Okay | 500 | 500 | 500 | 1100 | - |
| MH-Worse-Better | 500 | 500 | 500 | 1100 | - |

Table 5: **Evaluation Rollout Length by Dataset.** The table shows the evaluation rollout horizon for each dataset. These were determined based on the average rollout length of trajectories in each dataset.

## B.1 Data Collection

**Machine-Generated (MG) Datasets.** We trained RL agents to solve tasks and we subsequently collected data from the trained agents to form our MG datasets. We only considered the Lift and Can tasks, as the other tasks were exceedingly difficult for RL to solve. Our RL algorithm is based on the Soft Actor-Critic [30] implementation in RLkit. We trained the algorithm with episode lengths of 150 and a batch size of 1024, and used dense rewards to facilitate exploration. We trained on the Lift and Can tasks for 2.4 million and 7.2 million environment steps respectively. For each task we saved agent checkpoints every 600k timesteps in training, which amounts to 5 checkpoints for Lift and 13 checkpoints for Can. Most checkpoints achieved 0% average task success rate but the last few checkpoints reached an average task success rate of $\sim 70 - 80\%$. For generating the datasets, we loaded each saved checkpoint and generated 300 rollouts of a fixed length of 150 timesteps. We annotated the transitions in the rollouts with sparse task completion reward and a "done" signal of `True` when the corresponding next state in the transition represented task success. Overall the Lift dataset consists of 225k transitions, and the Can dataset consists of 585k transitions.

**Proficient-Human (PH) and Multi-Human (MH) Datasets.** Datasets were collected by using the RoboTurk platform [15, 17]. Our six human operators used smartphones to control robot arms hosted on a simulation server and were provided with video streams in a local web browser. The six operators were located at distances ranging from tens to several thousands of miles from the simulation server, and consisted of two "better" quality operators, two "okay" operators, and two "worse" operators. Since the Transport task requires controlling two arms, we used Multi-Arm RoboTurk [36] to allow pairs of operators to collect data. The Transport (PH) dataset consists of 300 demonstrations collected jointly by the two proficient operators, while the Transport (MH) dataset consists of 6 sets of 50 demonstrations, where each set consists of a pairing of demonstrators (Better-Better, Okay-Okay, Worse-Worse, Worse-Better, Okay-Better, Worse-Okay). As explained in Sec 4.2, we also further split the MH datasets into smaller subsets to investigate how suboptimal human data affects performance.

Unlike the other tasks, all MH data subsets consist of exactly 50 demonstrations, corresponding to a specific pairing of demonstrators. Table 4 shows the average trajectory length by dataset, which is a proxy for the quality of the dataset – less experienced operators produced longer trajectories.

**Can-Paired Dataset.** A single experienced operator collected 2 demonstrations for each of 100 task initializations on the Can task, resulting in 200 total demonstrations. Each pair of demonstrations consists of a "good" trajectory, where the can is picked up and placed in the correct bin, and a "bad" trajectory, where the can is picked up, and tossed outside of the robot workspace. Since the task initializations are identical, and the first part of each trajectory leading up to the can grasp is similar, there is a strong expectation for algorithms that deal with suboptimal data, to be able to filter the good trajectories from the bad ones, and achieve near-perfect performance (for reference, BC-RNN can achieve near-perfect performance on the 50% subset of Can (PH), which corresponds to 100 good quality demos, see Fig 3).

**Preparing Demonstration Subsets.** In order to prepare smaller datasets used in Sec 4.6 and Fig 3, we sampled a fixed portion (20% and 50%) of the trajectories uniformly per human that provided data. This ensured that the smaller size datasets were not biased towards higher or lower quality data. We also split all datasets and data subsets into training (90%) and validation (10%) subsets, using the same methodology. Models were not trained on the validation subsets.

## B.2 Training Setup

As mentioned in Sec 3, each agent is trained for $N$ epochs, where each epoch consists of $M$ gradient steps, and evaluated every $E$ epochs, by running 50 rollouts in the environment and reporting the success rate over a maximum horizon. All networks are trained using Adam optimizers [78]. The average trajectory length in each dataset (Table 4) was used to determine an appropriate evaluation rollout horizon for each dataset (Table 5). For each agent, we report the maximum success rate over the coarse of training, and average over 3 seeds. For agents trained with low-dimensional observations, $N = 2000$, $M = 100$, and $E = 50$, and for image observations, $N = 600$, $M = 500$, and $E = 20$. With the exception of the MG datasets, agents were only trained over 90% training subsets, with 10% held out as validation.

## C Problem Setup and Algorithm Details

Consider a robot manipulation task, formulated as an infinite-horizon discrete-time Markov Decision Process (MDP), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(\cdot|s,a)$ is the state transition distribution, $R(s,a,s')$ is the reward function, $\gamma \in [0,1)$ is the discount factor, and $\rho_0(\cdot)$ is the initial state distribution. At every step, an agent observes the state $s_t$, uses a policy $\pi$ to choose an action, $a_t = \pi(s_t)$, and observes the next state, $s_{t+1} \sim \mathcal{T}(\cdot|s_t,a_t)$, and reward, $r_t = R(s_t,a_t,s_{t+1})$. The goal is to learn an policy $\pi$ that maximizes the expected return: $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t,s_{t+1})]$. In this study, we assume access to an offline dataset of trajectories $\mathcal{D} = \{(s_0^i, a_0^i, r_0^i, s_1^i, ..., s_{T^i}^i)\}_{i=1}^N$ [12] and that a policy $\pi_\theta$ must be learned offline, without collecting any additional samples from the MDP.

**Rewards and Dones.** Unless otherwise mentioned, the rewards used in this study are binary task completion rewards, $R(s,a,s') = \mathbb{1}[s' \in \mathcal{G}]$, where $\mathcal{G} \subset \mathcal{S}$ is the set of all states where the task is considered to be solved. Similarly, the done signal, which indicates the end of an episode, is considered to be true for a transition $(s,a,r,s')$ if the task is solved in state $s'$ or if it is the last transition in a dataset trajectory. The reward and done signals are only used by Batch (Offline) RL algorithms (BCQ, CQL, IRIS).

We now outline the Imitation Learning and Batch (Offline) Reinforcement Learning algorithms used in our study.

### C.1 Behavioral Cloning (BC, BC-RNN, HBC)

Behavioral Cloning [8] (BC) is a common method for learning a policy from a set of demonstrations. It trains a policy $\pi_\theta(s)$ to clone the actions in the dataset via the objective:

$$\arg\min_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}} ||\pi_\theta(s) - a||^2.$$

BC-RNN is a variant of BC that uses a Recurrent Neural Network (RNN) as the policy network – this allows the policy to model temporal dependencies in the data through the recurrent hidden state. The network is trained on length-$T$ temporal sequences of data $(s_t, a_t, ..., s_{t+T}, a_{t+T})$. The network predicts the sequence of actions using the sequence of states as input. At test-time, the RNN policy network is unrolled one-step at a time, $a_t, h_{t+1} = \pi_\theta(s_t, h_t)$ where $h$ is the RNN hidden state. The hidden state is refreshed every $T$ steps.

Hierarchical Behavioral Cloning (HBC) trains hierarchical policies and has been shown to be effective in learning from offline human demonstrations [20, 10, 36]. HBC consists of a low-level policy that is conditioned on future observations $s_g \in \mathcal{S}$ (termed *subgoals*) and outputs action sequences to try and achieve them, and a high-level policy that predicts future subgoals from the current observation. The architecture and training procedure for the low-level policy $\pi_\theta^L(s, s_g)$ is nearly identical to BC-RNN – the only difference is the subgoal conditioning (during training, this is the final observation of the sampled sequence). The high-level policy $\pi_\theta^H(s)$ is trained to predict subgoal observations $s_{t+T}$ that occur $T$ timesteps in the future from the current observation $s_t$, and is often a conditional Variational Autoencoder (cVAE) [79] that learns a conditional distribution $\pi^H(s_{t+T}|s_t)$. At test-time, the high-level policy is queried for a new subgoal every $T$ timesteps, and the low-level policy is unrolled subsequently for $T$ timesteps using the predicted subgoal.

### C.2 Batch Constrained Q-Learning (BCQ)

Batch Constrained Q-Learning (BCQ) [26] is a commonly used algorithm for batch (offline) reinforcement learning [11, 12]. It maintains a Q-network $Q_\psi(s,a)$, a generative action model $p_\omega(a|s)$ (in the original implementation, this is a cVAE [79]), and (optionally) a perturbation actor network $\pi_\theta(s,a)$. Fujimoto et al. [26] noted that target value estimation in batch RL can suffer from overestimation error due to querying the Q-network on actions unseen in the dataset. To address this, BCQ modifies the way target value estimates are constructed for the temporal difference Q-network loss by approximately constraining the Q-network maximization to actions seen in the dataset.

To form the target value estimate, actions are sampled using the generative model and perturbed using the perturbation actor $A = \{a_i + \pi_\theta(s, a_i) | a_i \sim p_\omega(\cdot|s)\}_{i=1}^N$, and then used to maximize the Q-network at the next state $Q_{\text{target}} = r + \gamma \max_{a_i \in A} Q'_\psi(s', a_i)$. The Q-network is trained by minimizing the

temporal difference loss $(Q_\psi(s,a) - Q_{\text{target}})^2$. As in DDPG [80], the perturbation actor is trained to maximize Q-values via the loss $-Q_\psi(s, a + \pi_\theta(s,a))|a \sim p_\omega(\cdot|s)$. Using the perturbation actor to modify the samples is optional – we find that removing it is often beneficial (see Appendix I), as in other prior work [42]. At test-time, $N$ actions are sampled from the generative action model, perturbed by the actor (if present), and the action with the highest Q-value is selected.

## C.3 Conservative Q-Learning (CQL)

Conservative Q-Learning (CQL) [27] is a recent batch (offline) RL algorithm that addresses the overestimation issue of Q-values directly. While other offline RL algorithms place constraints on the policy to stay within the support of data, CQL places an implicit constraint on the Q-function that lower-bounds its values. Specifically a Q-value regularizer is added to the policy evaluation objective to ensure that the estimated Q-values under the policy $\pi_\theta(s,a)$ do not overestimate the Q-values under the data distribution $\mu(a|s)$:

$$Q^{k+1} \leftarrow \underset{Q}{\text{argmin}} \frac{1}{2}(Q(s,a) - Q_{\text{target}})^2 + \alpha\left( \mathbb{E}_{s\sim\mathcal{D},a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta(a|s)}[Q(s,a)] \right)$$

Kumar et al. [27] proved that this formulation lower bounds the true Q function and has theoretical improvement guarantees. CQL is simple to implement and has shown state-of-the-art empirical results on various offline datasets.

## C.4 Implicit Reinforcement without Interaction at Scale (IRIS)

Implicit Reinforcement without Interaction at Scale (IRIS) [20] is a batch (offline) RL algorithm proposed for learning from large robot manipulation datasets collected by multiple humans. It is identical to Hierarchical Behavioral Cloning (HBC) except that the high-level policy consists of both a cVAE subgoal sampler and a value function trained using Batch Constrained Q-Learning (BCQ). Similar to BCQ, the high-level policy selects the subgoal by sampling $N$ subgoals from the cVAE, and picking the state with the highest value estimate. Since the low-level policy is unimodal, all modeling of suboptimal and diverse data takes place in the high-level policy, at a reduced frequency (every $T$ timesteps), enabling temporal abstraction.

# D Hyperparameters

## D.1 Network Architecture

Here we briefly overview the general architecture design. Please refer to later sections for more detailed hyperparameter choices.

**General Network Details.** All Multi-Layer Perceptrons (MLPs) use ReLU activations. All Recurrent Neural Networks (RNNs) are 2-layer LSTMs, where the final layer hidden states are fed to downstream modules.

**Encoding Observations.** All networks have an observation encoder that processes observation dictionaries into a single vector. The encoder takes image observations, passes each through an observation-specific encoder into a low-dimensional vector, and finally concatenates the encoded image vectors with the low-dimensional observation vectors. For example, visuomotor policies typically contain two image encoders, one for the frontview camera, and one for the wrist camera. Each image encoder consists of a ResNet-18 network [46] followed by a spatial-softmax layer [45].

**Observation-Conditioned Network Structure.** Here we describe the general structure of networks that take observations as inputs. This includes policies, as well as value and state-action value functions, and comprises the majority of all networks used by the algorithms. First, observations are encoded (if they are images) and concatenated together into a flat vector. Next, in the case of networks that use an RNN (policies for BC-RNN, HBC, and IRIS), the flat encoded observations are sent through the RNN in order to produce hidden state outputs. Finally, outputs are passed to an MLP consisting of one or more layers in order to predict the item of interest. In the case of policy networks, the output either consists of raw actions, or the parameters of an action distribution (e.g. Gaussian Mixture Model parameters), while value functions output a single scalar. Policy action predictions (raw action predictions and mean parameter predictions) are passed through a tanh layer for normalization to [-1, 1].

## D.2 Hyperparameter Selection Procedure

For each algorithm, we tuned hyperparameters separately for the Machine-Generated (MG) datasets, the Proficient-Human (PH) datasets, and the Multi-Human (MH) datasets. We used both the Lift (MG) and Can (MG) datasets for selecting a single set of hyperparameters for use on all MG datasets. We used the Square (PH) and Transport (PH) datasets for selecting a single set of hyperparameters for use on all PH datasets. We used the Square (MH) and Transport (MH) datasets for selecting a single set of hyperparameters for use on all MH datasets. We used Weights and Biases [81] to conduct hyperparameter tuning.

Note that the Tool Hang (sim) and all real tasks were purely used for evaluation purposes – **no hyperparameter tuning took place on these tasks**. This was to see whether our insights from hyperparameter tuning above could transfer to our hardest simulation task, and directly to real robot datasets.

Also note that we excluded HBC and IRIS from image-based training, due to the dependence of these algorithms on subgoal reconstructions, which could be problematic for high-dimensional images.

**BC and BC-RNN.** We scanned the following hyperparameters for BC and BC-RNN:

- **Learning rate**: We compared $1e-3$ and $1e-4$ for both BC and BC-RNN. We found lower learning rate to perform better consistently.

- **Actor network dimension**: We scanned different dimensions for the action output network (MLP): $[300, 400]$, $[1000, 1000]$, and no MLP (directly output from RNN). We found higher capacity works better for BC, and no MLP works better for RNN. Our hypothesis is that RNN already has enough capacity to learn the tasks, and larger actor network results in overfitting.

- **GMM for action output**: Stochastic policy (GMM output) performs better than deterministic policy, although the gap is smaller for BC-RNN than for BC. We scanned different number of modes: $\{5, 10, 100\}$ standard deviation minimum clipping: $\{1e-2, 1e-4, 1e-6\}$ and observed no significant difference in performance.

- **Sequence length (RNN)**: We compared sequence length $\{10, 30, 50\}$. We found longer sequence length does not improve performance significantly. We opt for short sequence length for training efficiency.
- **RNN Hidden Dim**: We compared different hidden dimension sizes for the RNN (LSTM): $\{100, 400, 1000, 2000\}$. We observed that 400 and 1000 tend to work well for most tasks.

**BCQ.** As in the original implementation [26], we used two critic networks. When using the perturbation actor, we also left the scale unchanged from the original (0.05). When using the VAE action sampler, we used a latent dimension of 14. We scanned the following hyperparameters:

- **Learning rate**: For the learning rate of the critic (CLR) and the action sampler (ASLR), we compared $1e-3$ and $1e-4$. We found smaller learning rate for the action sampler usually has better performance on human datasets, while there was not much difference between the two learning rates for the critic.
- **Actor/Critic network dimension**: We compared the networks with layer dimensions $[300, 400]$ and $[1024, 1024]$. We found there is no difference for them with Machine-Generated (MG) and Proficient-Human (PH) datasets. For Multi-Human (MH) dataset, larger network dimension $[1024, 1024]$ has better results.
- **Perturbation actor**: We compare the results of whether use the perturbation actor for the BCQ action sampling[80]. As is shown in Table. 18, we found the performance drastically decreases when using the actor. By contrast, with the machine-generated data (MG), we found BCQ with actor enabled usually has better performance.
- **Action sampler (VAE vs GMM)**: We compared two types of action sampler - the VAE and GMM and found that the VAE generally has better performance than the GMM sampler. However, the GMM sampler also allows for a direct comparison with BC – see Table 19 for results.
- **VAE KL**: We scanned $\{5e-1, 5e-2, 5e-3, 5e-4\}$ the weight of KL for the VAE action sampler and found that larger KL weights $5e-1, 5e-2$ show better performance.
- **VAE Layer Dims:**: For the VAE action sampler, we compared the encoder / decoder / prior layer dimensions $[300, 400]$ and $[1024, 1024]$. We found there is no significant difference for them in low-dimensional tasks. With image observation input, larger dimension $[1024, 1024]$ outperforms $[300, 400]$.
- **VAE prior**: While the VAE prior is commonly a normal prior $N(0, 1)$, it can also be learned as part of the KL loss. We compared using the normal prior to using a state-dependent GMM prior, whose parameters are output by an MLP. We found that the normal prior $N(0, 1)$ generally performs better than the learned GMM prior, except for the Multi-Human(MH) dataset. Therefore, we opted the learned GMM prior only for Multi-Human(MH) dataset and keep $N(0, 1)$ prior for the others.
- **Tau (target network update rate)**: We compared $5e-3$ and $5e-4$ and found the lower value of $5e-4$ to give better results.
- **Num action samples (train/test)**: During training, BCQ requires generating a number of samples from the action sampler to do the Bellman backup, while at test-time, BCQ requires generating action samples to approximately maximize the critic over the samples to choose an appropriate action. We compared $[10, 100]$ and $[100, 1000]$ for the number of action samples during training and testing and found that $[10, 100]$ works better.

**CQL.** In contrast to other algorithms, we primarily tuned CQL hyperparameters on the low-dim Lift (MG) dataset, due to poor performance on the harder Square (PH) and Transport (PH) datasets. In subsequent experiments, we found that these hyperparameter settings worked better than other choices on other datasets as well. For all experiments we used a discount factor of $\gamma = 0.99$, a target network update rate of $\tau = 0.005$, and actor / critic network layer sizes of $[300, 400]$. We scanned the following hyperparameters:

- **Learning rate**: For low-dim experiments we scanned an extensive sweep of learning rates for the Q network and the policy, spanning values of $\{1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$. For the Q network we found that $1e-3$ performed best — lower values slowed down learning while higher values led to unstable learning. For the policy we found

that $3e - 4$ and $1e - 3$ worked well, though $3e - 4$ performed slightly better. For image experiments we performed a smaller sweep and found that $1e - 4$ for both the Q network and policy performed relatively well compared to other learning rates.

- **Deterministic backup**: As suggested by the CQL implemented from Kumar et al. [27], we experimented with a deterministic Bellman backup objective — i.e. removing the additional entropy term from the target Q value backup. We found that the deterministic backup outperformed the non-deterministic backup, though at times the gains were marginal. We subsequently chose to deterministic variant.

- **BC start steps**: We experimented with replacing the policy loss objective for CQL with the behavior cloning objective for the first 40000 gradient steps of training. We found that the performance of the agent degraded after these initial gradient steps, so we set this hyperparameter to 0 in all subsequent experiments.

- **Batch size**: For low-dim experiments, we found that increasing the batch size from the default value of 100 to 1024 can lead to significant gains in stability and performance. For image experiments we were bottlenecked by GPU memory and we used a batch size of 8.

- **Lagrange variant**: We found that the Lagrange variant consistently performed better than the non-Lagrange variant and was less sensitive to hyperparameters. We therefore decided to use the Lagrange variant in all of our experiments.

- **Lagrange threshold $\tau$**: We found that a threshold of $\tau = 1$ often caused the dual weight to diverge to large values. We subsequently experimented with higher values of $5, 10, 25$ and found that the algorithm was relatively stable with all of these with no major difference in performance. We subsequently chose $\tau = 5$.

**HBC.** We scanned the following hyperparameters for HBC:

- **Learning Rate:** We scanned $1e - 3, 1e - 4$ for both the policy and goal learning rate, and generally found the higher learning rates to perform better.

- **VAE vs AE:** We compared using a VAE vs. only an AE for the planner network, and found the best performing VAE model outperformed the best performing AE model.

- **VAE KL:** We compared using $5e - 1, 5e - 2, 5e - 3, 5e - 4$, and found that $5e - 4$ worked the best.

- **VAE Prior:** We compared using $\mathcal{N}(0, 1)$, Learned GMM priors, and found that, when tuned, the GMM prior worked the best.

- **VAE Latent Dim:** We compared $2, 16, 100$, and found that $16$ worked the best.

- **VAE Layer Dims:** We compared $[300, 400], [1024, 1024]$ as the encoder / decoder / prior layer dimensions, and found that $[1024, 1024]$ worked the best.

- **RNN Hidden Dim:** We compared $100, 400, 1000, 2000$ and found that $400$ worked the best.

- **Actor MLP Dims:** We compared $[], [300, 400], [1024, 1024]$ and found that no hidden layers ($[]$) worked the best.

**IRIS.** We first initialize our scan with the best hyperparameters from HBC and BCQ. Note that from our BCQ scan, we use **Action Sampler Learning Rate** $= 1e - 4$ and **Num Action Samples (Train/Test)** $= 10/100$. We also use **Tau** $= 5e - 4$ and no value actor for PH and MH datasets and **Tau** $= 5e - 3$ with value actor enabled for MG datasets. We then proceeded to scan over the following hyperparameters:

- **Learning Rate:** We compared $1e - 3, 1e - 4$ for the policy, goal, and value learning rates individually, and generally found the higher learning rates to perform better. The exception is the multi-human setting, where we found a lower Value LR to work better.

- **Value KL Weight:** We compared $0.5, 0.05$, and found that both values generally performed similarly.

- **Value Actor:** We compared using a value actor True, False on the MG datasets, and found that an actor can improve results on the Can MG dataset.

## D.3 Final Hyperparameters

We present our finalized set of hyperparameters in Tables 6 and 7 for BC, Tables 8 and 9 for BC-RNN, Tables 10 and 11 for BCQ, Table 12 for HBC, Table 13 for IRIS, and Tables 14 and 15 for CQL. Each column shows the hyperparameters for learning from a dataset setting (shared across environments): PH for proficient human, MH for multiple humans, MG for machine-generated. - means the hyperparameter is inherited from the default hyperparameters shown on the left-most column. For further details on the training setup shared by all algorithms (such as the number of gradient steps and epochs), see Appendix B.2.

Table 6: **BC Hyperparameters - Low-Dim (LD)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| LR | $1e-4$ | - | - | $1e-3$ |
| Actor MLP Dims | $[1024, 1024]$ | - | - | - |
| GMM Num Modes | 5 | - | - | no-gmm |

Table 7: **BC Hyperparameters - Image (IM)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| LR | $1e-4$ | - | - | - |
| Actor MLP Dims | $[1024, 1024]$ | - | - | - |
| GMM Num Modes | 5 | - | - | no-gmm |
| Image Encoder | ResNet-18 | - | - | - |
| SpatialSoftmax [45] (num-KP) | 64 | - | - | - |

Table 8: **BC-RNN Hyperparameters - Low-Dim (LD)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| LR | $1e-4$ | - | - | - |
| Actor MLP Dims | [] | - | - | - |
| RNN Hidden Dim | 400 | - | - | - |
| RNN Seq Len | 10 | - | - | - |
| GMM Num Modes | 5 | - | - | no-gmm |

Table 9: **BC-RNN Hyperparameters - Image (IM)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| LR | $1e-4$ | - | - | - |
| Actor MLP Dims | [] | - | - | - |
| RNN Hidden Dim | 1000 | - | - | - |
| RNN Seq Len | 10 | - | - | - |
| GMM Num Modes | 5 | - | - | no-gmm |
| Image Encoder | ResNet-18 | - | - | - |
| SpatialSoftmax [45] (num-KP) | 64 | - | - | - |

## D.4 Additional Details on Hyperparameter Choice Study for BC-RNN

In Sec 4.4, Fig 2b, and Fig 2c, we presented results that showcase how changing a subset of BC-RNN hyperparameters can result in large performance decreases. In this section, we provide more details for each change.

**Larger LR.** We changed the policy learning rate from the default 1e-4 to 1e-3.

Table 10: **BCQ Hyperparameters - Low Dim (LD)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| CLR | $1e-4$ | - | - | $1e-3$ |
| ASLR | $1e-4$ | - | - | $1e-3$ |
| Critic Dims | $[300, 400]$ | - | $[1024, 1024]$ | - |
| Actor Dims | $[300, 400]$ | - | $[1024, 1024]$ | - |
| Perturb Actor | False | - | - | True |
| Action Sampler | VAE | - | - | - |
| VAE KL | $5e-2$ | - | $5e-1$ | $5e-1$ |
| VAE Dims | $[300, 400]$ | - | $[1024, 1024]$ | - |
| VAE Prior | $N(0,1)$ | - | GMM | - |
| Tau | $5e-4$ | - | - | $5e-3$ |
| Num Action Samples | $[10, 100]$ | - | - | - |

Table 11: **BCQ Hyperparameters - Image (IM)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| CLR | $1e-4$ | $1e-3$ | - | $1e-3$ |
| ASLR | $1e-4$ | - | - | $1e-3$ |
| Critic Dims | $[300, 400]$ | - | $[1024, 1024]$ | - |
| Actor Dims | $[300, 400]$ | - | $[1024, 1024]$ | - |
| Perturb Actor | False | - | - | - |
| Action Sampler | VAE | - | - | - |
| VAE KL | $5e-2$ | - | - | $5e-1$ |
| VAE Dims | $[1024, 1024]$ | - | - | - |
| VAE Prior | $N(0,1)$ | - | GMM | - |
| Tau | $5e-4$ | - | - | $5e-3$ |
| Num Action Samples | $[10, 100]$ | - | - | - |

Table 12: **HBC Hyperparameters - Low Dim (LD)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| Planner LR | $1e-3$ | - | - | - |
| Planner VAE KL | $5e-4$ | - | - | - |
| Planner VAE GMM Prior | True | - | - | - |
| Planner VAE GMM Latent Dim | 16 | - | - | - |
| Planner VAE MLP Dims | $[1024, 1024]$ | - | - | - |
| Actor LR | $1e-3$ | - | - | - |
| Actor RNN Hidden Dim | 400 | - | - | 100 |
| Actor MLP Dims | $[]$ | - | - | $[1024, 1024]$ |

Table 13: **IRIS Hyperparameters - Low Dim (LD)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| Planner LR | $1e-3$ | - | - | - |
| Planner VAE KL | $5e-4$ | - | - | - |
| Planner VAE GMM Prior | True | - | - | - |
| Planner VAE GMM Latent Dim | 16 | - | - | - |
| Planner VAE MLP Dims | $[1024, 1024]$ | - | - | - |
| Actor LR | $1e-3$ | - | - | - |
| Actor RNN Hidden Dim | 400 | - | - | - |
| Actor MLP Dims | $[]$ | - | - | - |
| Value LR | $1e-3$ | - | $1e-4$ | - |
| Value KL | 0.5 | - | 0.05 | - |

Table 14: **CQL Hyperparameters - Low Dim (LD)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| Q network LR | $1e-3$ | - | - | - |
| Policy LR | $3e-4$ | - | - | - |
| Deterministic Backup | True | - | - | - |
| BC Start Steps | 0 | - | - | - |
| Batch Size | 1024 | - | - | - |
| Lagrange | True | - | - | - |
| Lagrange Threshold $\tau$ | 5.0 | - | - | - |
| Actor MLP Dims | $[300, 400]$ | - | - | - |

Table 15: **CQL Hyperparameters - Image (IM)**

| Hyperparameter | Default | Dataset | | |
| --- | --- | --- | --- | --- |
| | | PH | MH | MG |
| Q network LR | $1e-4$ | - | - | - |
| Policy LR | $1e-4$ | - | - | - |
| Deterministic Backup | True | - | - | - |
| BC Start Steps | 0 | - | - | - |
| Batch Size | 8 | - | - | - |
| Lagrange | True | - | - | - |
| Lagrange Threshold $\tau$ | 5.0 | - | - | - |
| Actor MLP Dims | $[300, 400]$ | - | - | - |

**No GMM.** By default, all BC-RNN policies on PH and MH learned a Gaussian Mixture Model (GMM) distribution. Here, we replace the GMM distribution with a direct action prediction.

**Larger MLP.** As noted in Appendix D.1 above, there is an MLP that transforms RNN hidden states into action (or action distribution) predictions. By default, we use a single linear layer, but here we try adding two layers of size 1024.

**Shallow Conv.** As noted in Appendix D.1 above, all image encoders are ResNet-18 networks. Here, we tried replacing the ResNet with the shallow convolutional network from Finn et al. [45].

**Smaller RNN Dim.** By default, we use a hidden layer size of 400 for low-dimensional datasets, and 1000 for image datasets. Here, we tried reducing the dimension to 100 for low-dim, and 400 for image.
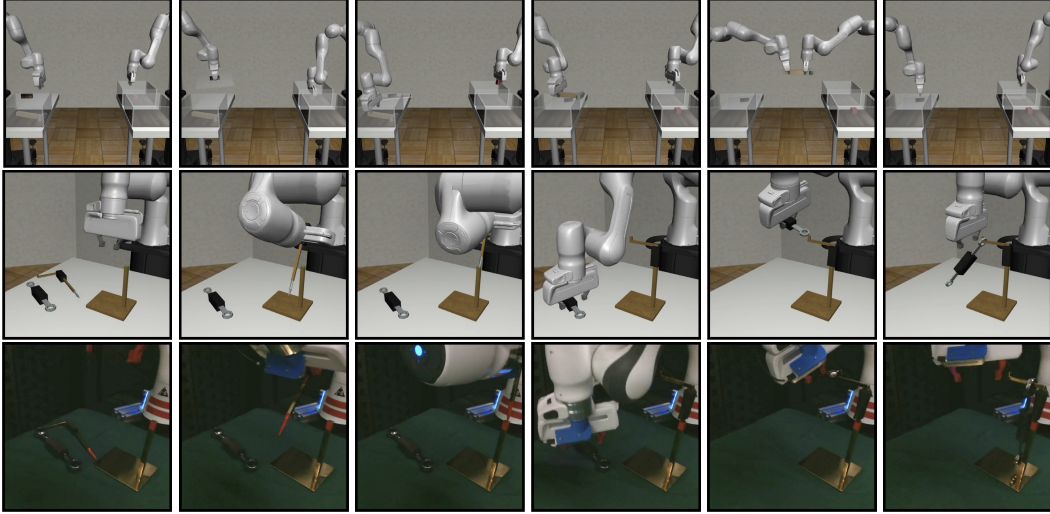
# E    Additional Details on Task Environments



Figure E.1: **Task Demonstrations.** We showcase example demonstration trajectories for the Transport (top), Tool Hang (middle), and Tool Hang (Real) (bottom) tasks, to provide a better sense of each stage of these tasks.

## E.1    Simulation Tasks

All simulation tasks were designed using MuJoCo [82] and the robosuite framework [76]. We used Panda robotic arms in both simulation and the real world for this study. The action space for the agent is a 7-dimensional vector for each arm where the first 3 coordinates are the desired translation from the current end effector position, the next 3 coordinates encoder the desired delta rotation from the current end effector rotation, and the final coordinate controls the opening and closing of the gripper fingers. The delta rotation is encoded in axis-angle form, where the norm of the 3-vector is the angle, and normalizing the 3-vector produces the axis. The policy outputs actions at a rate of 20 Hz. Policy actions are transformed into end effector target poses and sent to an operational space controller [83] that outputs the robot joint torques at 500 Hz to try and achieve the desired cartesian poses.

All simulation environments will be released along with datasets and codebase upon publication. We next describe the low-dimensional object observations and initial state randomization for each task.

**Lift.** Object observations (10-dim) consist of the absolute cube position and cube quaternion (7-dim), and the cube position relative to the robot end effector (3-dim). The cube pose is randomized at the start of each episode with a random z-rotation in a small square region at the center of the table.

**Can.** Object observations (14-dim) consist of the absolute can position and quaternion (7-dim), and the can position and quaternion relative to the robot end effector (7-dim). The can pose is randomized at the start of each episode with a random z-rotation anywhere inside the left bin.

**Square.** Object observations (14-dim) consist of the absolute square nut position and quaternion (7-dim), and the square nut position and quaternion relative to the robot end effector (7-dim). The square nut pose is randomized at the start of each episode with a random z-rotation in a square region on the table.

**Transport.** Fig E.1 (top) shows a full demonstration of the task. Object observations (41-dim) consist of the absolute position and quaternion of the hammer (7-dim), the absolute position and quaternion of the trash cube (7-dim), the absolute position and quaternion of the lid handle (7-dim), the target bin position (3-dim), the trash bin position (3-dim), the relative positions of the hammer and the lid handle with respect to the first arm end effector (6-dim), the relative positions of the hammer and trash cube with respect to the second arm end effector (6-dim), a binary indicator for the hammer reaching the target bin (1-dim), and a binary indicator for the trash reaching the trash bin (1-dim). The position of all bins, the lid, the trash cube, and the hammer are randomized in small squares at the start of each episode. The z-rotation of the trash cube and the hammer are also randomized with a full range of 108 and 60 degrees respectively.

**Tool Hang.** Fig E.1 (middle) shows a full demonstration of the task. Object observations (44-dim) consist of the absolute position and quaternion and relative pose and quaternion with respect to the end effector of the base frame (14-dim), the insertion hook (14-dim), and the ratcheting wrench (14-dim), as well as binary indicators for whether the stand was assembled (1-dim) and whether the tool was successfully placed on the stand (1-dim). The position of the insertion hook and ratcheting wrench and z-rotation (range of 40 degrees) are randomized in a small square at the beginning of the episode.

## E.2 Real World Task Setup

We first describe details about the robot workspace and setup. Next, we discuss the materials needed to construct each physical task. We took care to approximately match the visual appearance, the physical dimensions, and the task initialization randomizations of the real tasks to those in simulation.

**Workspace and Setup.** Our physical robot workspace consists of a Franka Emika Panda robotic arm, a front-view Intel RealSense SR300 camera, and a wrist-mounted Intel RealSense D415 camera. The robot arm and the front-view camera are attached rigidly to the table, while the wrist-view camera points towards the space in between the robot gripper fingers. Demonstration data was collected from the robot sensors and the two cameras at approximately 20 Hz. Similar to simulation, the robot is controlled using an operational space controller, using the same action space as the one in simulation (see above).

**Lift (Real).** A white 3D-printed cube that measures 4 cm in all dimensions was used for this task, with a small initialization square that roughly corresponds to the one for the simulation task.

**Can (Real).** We purchased this tray for the left bin (where the can starts in each episode), and four of these small boxes to construct the right bin. We used a 7.5 oz Coca Cola Zero Sugar Diet Soda Can (empty, and stuffed with some paper) as the coke can.

**Tool Hang (Real).** Fig E.1 (bottom) shows a full demonstration of the task on the real robot. We purchased this handbag stand, and sawed the base rod and the hook rod in half. We also outfitted the bottom of the hook rod with a soldering iron tip using 2-part epoxy in order to make the hook more amenable to insertion. We also purchased this ratcheting wrench tool set and used the 17mm-19mm wrench for the task.
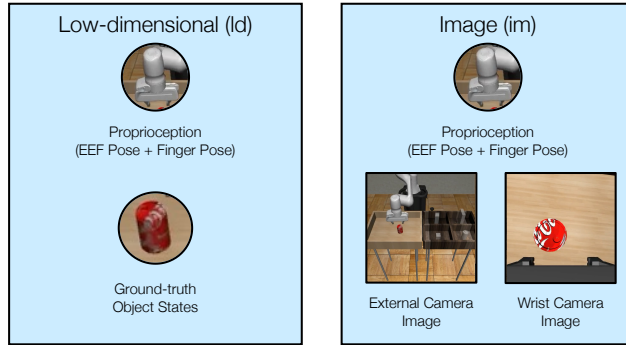
## E.3 Observation Space Details



Figure E.2: **Observation Spaces.** The figure shows the low-dimensional and image observation spaces used for our study. Proprioception, and camera images are provided for each robot arm in the environment.

In this section, we provide additional details related to the observation spaces used to train agents (shown in Fig E.2). Both low-dim agents and image agents receive proprioception observations (9-dim per arm) consisting of the end effector position (3-dim), quaternion (4-dim), and gripper finger positions (2-dim). The low-dim agents also receive object observations (described above, see Appendix E.1), while image agents receive an external camera image and a wrist camera image per robot arm. We first provide further details on cameras and image sizes used per task, then we discuss pixel shift randomization details (shown to be crucial for visuomotor learning, see Sec 4.3 and Fig 2a), and finally, we provide more details on the experiments in Sec 4.3.

**Image Observations per Task.** For all tasks except Transport, we provide two image observations – one from a front-view camera and one from a wrist-mounted camera. For the Transport task, we provide four image observations to the agent – two from shoulder-view cameras per arm, and two from wrist-mounted camera on each arm. The front-view and shoulder-view cameras are the same cameras used by human operators to provide task demonstrations. All simulation tasks, with the exception of Tool Hang, provide 84 by 84 images. All real robot tasks, with the exception of Tool Hang, provide 120 by 120 images. Tool Hang, in both simulation and real world provides 240 by 240 images (due to the need for high-precision control). On the real robot, raw camera frames are read from the camera at a full resolution of 640 by 480, then a center crop of 480 x 480 is applied, and finally, images are resized to the appropriate resolution.

**Additional Details for Pixel Shift Randomization.** To implement random pixel shifts [84–87] for input image observations, we take large random crops from the source images when feeding image observations to any network. For each input image of width $W$ and height $H$, we randomly crop a region of width $w$ and height $h$, where $W - w$ and $H - h$ is small. For $(H, W) = (84, 84)$ we use $(h, w) = (76, 76)$, for $(H, W) = (120, 120)$ we use $(h, w) = (108, 108)$ and for $(H, W) = (240, 240)$ we use $(h, w) = (216, 216)$.

**Additional Details for Observation Space Study.** Here, we describe the observations added for the experiments presented in Sec 4.3 and Fig 2a. When adding EEF Vel observations, we added linear end effector velocity (3-dim per arm), angular end effector velocity (3-dim per arm), and gripper finger velocities (2-dim per arm). When adding Joint observations, we encoded the joint positions using cosine (7-dim per arm) and sine (7-dim per arm), and also provided joint velocities (7-dim per arm).

# F  Learning Curves

In this section, we present learning curves that show success rate versus epoch for BC-RNN on the Proficient-Human (PH) and Multi-Human datasets. Notice that epoch-to-epoch performance can vary drastically, even though the number of evaluation rollouts per checkpoint is high (50), suggesting that this is caused by the mismatch between training and evaluation objectives (C4). See Appendix G for more discussion.



(a) Lift (PH, ld)   (b) Can (PH, ld)   (c) Square (PH, ld)   (d) Transport (PH, ld)

(e) Lift (MH, ld)   (f) Can (MH, ld)   (g) Square (MH, ld)   (h) Transport (MH, ld)

(i) Lift (PH, im)   (j) Can (PH, im)   (k) Square (PH, im)   (l) Transport (PH, im)

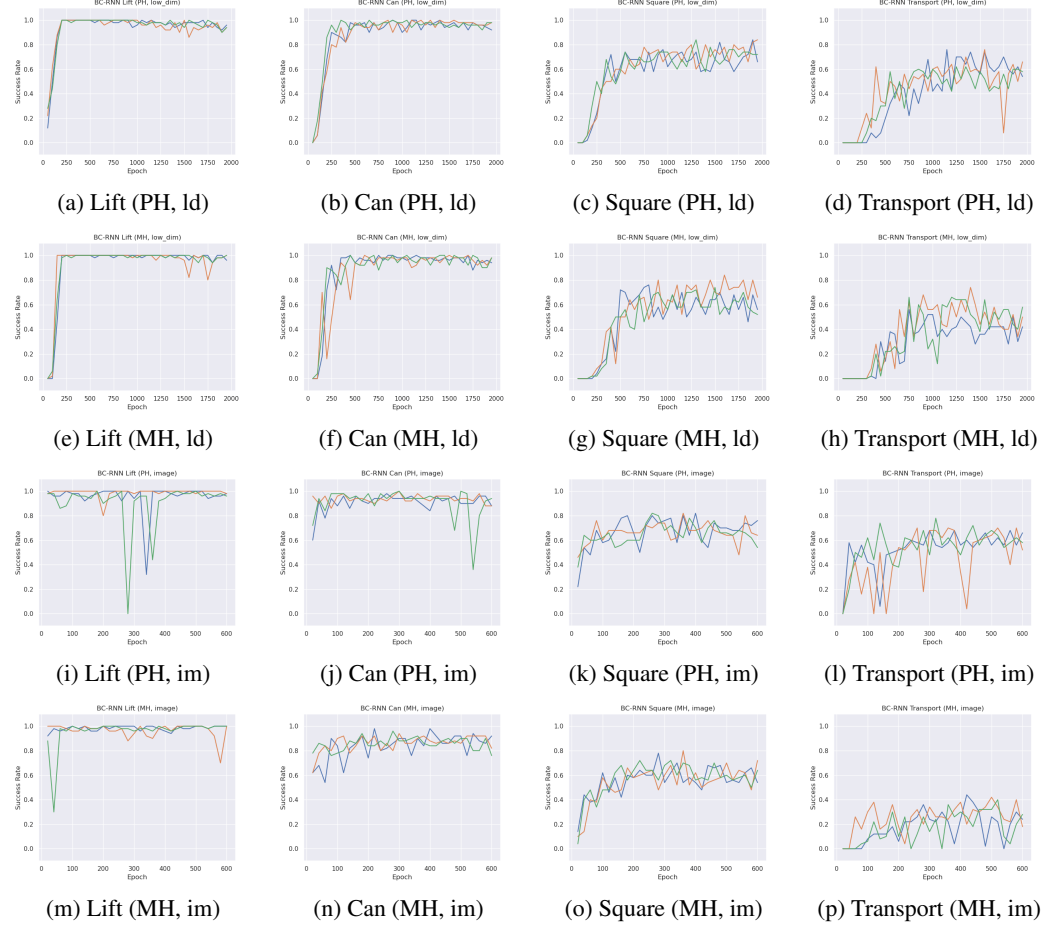(m) Lift (MH, im)   (n) Can (MH, im)   (o) Square (MH, im)   (p) Transport (MH, im)

Figure F.1: **Learning Curves.** We show the success rate versus epoch for BC-RNN on the Proficient-Human (PH) and Multi-Human (MH) datasets, across 3 seeds. Notice that epoch-to-epoch performance can vary drastically.

# G  Additional Results on Policy Selection

In this section, we present some more results and discussion on offline policy selection related to Sec 4.5 and Fig 4a. We first show that policy performance can vary substantially during a training run – this is why policy selection is non-trivial. One might expect that adding more validation data could help improve its use as a selection criteria for selecting a good policy. We empirically show that this is not necessarily true. Finally, we show that success rate can keep climbing, even while the validation loss increases substantially.

**Policy checkpoints can vary substantially in performance during training, even when performance appears to converge.** Fig F.1 shows several different learning curves for BC-RNN agents on our human datasets. Low-dimensional agents exhibit significant variance in policy success rate even in later stages of training, on harder tasks like Square (see Fig F.1c and Fig F.1g) and Transport (see Fig F.1d and Fig F.1h). While this is also true for image agents (see Fig F.1k and Fig F.1l), even simpler tasks like Lift (Fig F.1i) and Can (Fig F.1j) can suffer from such variance in performance. This kind of variance in performance is problematic for real world settings where it's not feasible to run 50 rollouts per checkpoint for each training run, as we have done in simulation. This makes offline policy selection a difficult, but important problem to solve.

**Increasing the amount of validation data does not improve policy selection using validation loss.** Fig 4a used a validation dataset size that was 10% of the collected data. We also tried training low-dim BC-RNN policies on the Square (PH) and Transport (PH) datasets, where we used 30% of the collected data for validation (and only 70% for training). Across 3 seeds, the best policy on Square (PH) achieves $80.7 \pm 0.9$, while the policy achieving the lowest validation loss achieves $2.7 \pm 1.9$, and the best policy on Transport (PH) achieves $64.0 \pm 2.8$ while the policy achieving the lowest validation loss achieves $0.7 \pm 0.9$.

**Success rate can increase even while validation loss increases substantially.** Empirically, we found that best validation loss occurs relatively early in training (epoch 100-300) but the best performance occurs much later. In Fig G.1, we present selected plots of success rate and validation loss versus epoch, to show this. The plots also show that validation loss can keep increasing substantially in later epochs – despite this, the success rate also keeps increasing. This further shows that validation loss is a poor measure of policy performance.



|  (a) Square (ld) – SR | (b) Square (ld) – Loss | (c) Transport (ld) – SR | (d) Transport (ld) – Loss |

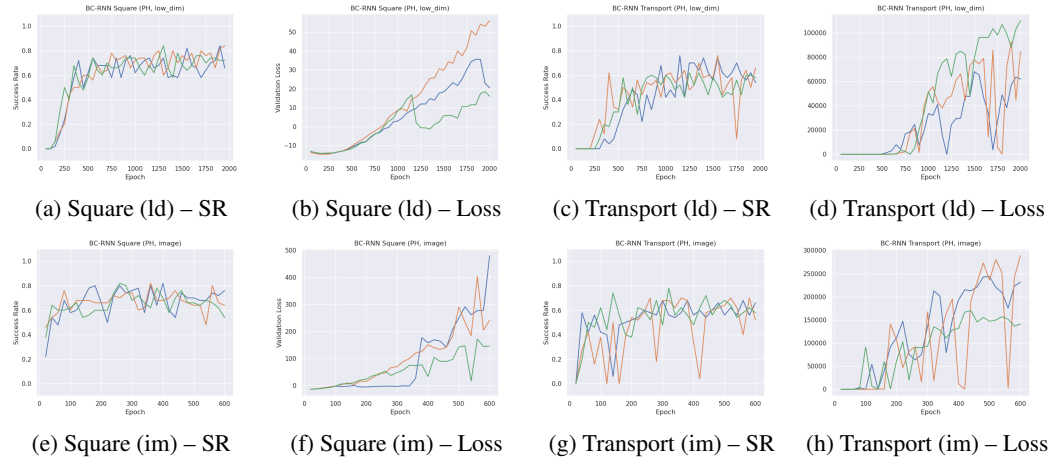|  (e) Square (im) – SR | (f) Square (im) – Loss | (g) Transport (im) – SR | (h) Transport (im) – Loss |

Figure G.1: **Success Rate and Validation Loss.** We show the success rate versus epoch and validation loss versus epoch side-by-side for BC-RNN on the Square (PH) and Transport (PH) datasets, across 3 seeds. The top row is low-dim observations, and the bottom is image observations. Notice that in many cases, the validation loss increases along with success rate.

# H Additional Image Dataset Results

## H.1 Machine-Generated Datasets

| Dataset | BC | BC-RNN | BCQ | CQL |
|---------|-----|--------|-----|-----|
| Lift (MG) | 79.3 ± 6.6 | 81.3 ± 5.7 | **88.7 ± 6.6** | 2.7 ± 0.9 |
| Can (MG) | 60.7 ± 2.5 | 63.3 ± 2.5 | **65.3 ± 2.5** | 0.0 ± 0.0 |

Table 16: **Machine Generated Results (image).** We present success rates averaged over 3 seeds for each method across the image Machine-Generated (MG) datasets. BCQ outperforms the other methods; however, it is possible that recent batch RL methods [38, 43] that have been shown to work on pixel observations might be able to perform even better.

Table 16 shows results on the Machine-Generated (MG) datasets with image observations. BCQ outperforms the other methods; however, it is possible that recent batch RL methods [38, 43] that have been shown to work on pixel observations might be able to perform even better.

## H.2 Suboptimal Human Datasets

| Dataset | BC | BC-RNN | BCQ | CQL |
|---------|-----|--------|-----|-----|
| Can-Worse | 54.7 ± 2.5 | **70.0 ± 3.3** | - | - |
| Can-Okay | 85.3 ± 0.9 | **90.0 ± 3.3** | - | - |
| Can-Better | **96.0 ± 0.0** | **96.0 ± 2.8** | - | - |
| Can-Worse-Okay | 72.7 ± 1.9 | **94.0 ± 1.6** | - | - |
| Can-Worse-Better | 84.0 ± 2.8 | **92.7 ± 1.9** | - | - |
| Can-Okay-Better | 94.7 ± 0.9 | **98.0 ± 0.0** | - | - |
| Square-Worse | 17.3 ± 1.9 | **36.7 ± 0.9** | - | - |
| Square-Okay | 28.7 ± 5.0 | **44.0 ± 1.6** | - | - |
| Square-Better | 49.3 ± 1.9 | **60.0 ± 2.8** | - | - |
| Square-Worse-Okay | 28.7 ± 2.5 | **52.7 ± 6.2** | - | - |
| Square-Worse-Better | 38.7 ± 0.9 | **57.3 ± 3.4** | - | - |
| Square-Okay-Better | 47.3 ± 2.5 | **62.0 ± 5.7** | - | - |
| Can-Paired | 56.7 ± 0.9 | **62.0 ± 2.8** | 50.0 ± 5.9 | 0.0 ± 0.0 |

Table 17: **Results on Suboptimal Human Data (Image).** We present success rates averaged over 3 seeds for each method across different subsets of the Multi-Human datasets, corresponding to mixtures of demonstrations from "Better", "Adequate", and "Worse" human operators, and finally on a diagnostic dataset with paired success and failure human trajectories for each starting initialization. We omitted Batch RL methods (except for Can-Paired) due to their poor performance on the other human datasets with image observations.

Table 17 shows results on our multi-human data subsets with image observations. We excluded batch RL methods due to their poor performance on human datasets with image observations. BC-RNN improves over BC on all datasets, especially datasets with lower quality data.

# I Additional Batch (Offline) RL Results

## I.1 Hyperparameter Sensitivity

### I.1.1 BCQ

| Dataset | Default | Perturbation Actor |
|---|---|---|
| Lift (PH) | $100.0 \pm 0.0$ | $72.0 \pm 4.3$ |
| Can (PH) | $88.7 \pm 0.9$ | $8.0 \pm 4.3$ |
| Square (PH) | $50.0 \pm 4.9$ | $3.3 \pm 0.9$ |
| Transport (PH) | $7.3 \pm 3.3$ | $0.0 \pm 0.0$ |

Table 18: **BCQ Hyperparameter Sensitivity - Actor.** The perturbation actor causes large performance drops on human datasets.

| Dataset | Default (BCQ) | Default (BC) | BCQ (BC param) |
|---|---|---|---|
| Can (PH) | $88.7 \pm 0.9$ | $95.3 \pm 0.9$ | $32.0 \pm 1.6$ |
| Square (PH) | $50.0 \pm 4.9$ | $78.7 \pm 1.9$ | $22.7 \pm 6.6$ |
| Can (MH) | $62.7 \pm 8.2$ | $86.0 \pm 4.3$ | $12.0 \pm 2.8$ |
| Square (MH) | $14.0 \pm 4.3$ | $52.7 \pm 6.6$ | $4.0 \pm 0.0$ |

Table 19: **BCQ Hyperparameter Sensitivity - matching parameters to BC.** We find that matching the hyperparameters of the BCQ action sampler to the ones we used for BC is not sufficient to improve performance.

In Table 18, we show that using the BCQ perturbation actor (see Appendix C.2) can have a catastrophic effect when training on human datasets. The results show that there is a large performance drop after enabling the perturbation actor (over 80% on Can-PH, for example). This result showcases BCQ is highly sensitive to the perturbation actor.

In Sec 4.1 and Sec 4.2, we empirically saw BCQ consistently underperform compared to BC. To further investigate this problem, we tried matching the hyperparameters of the BCQ action sampler to the BC model we used (the same learning rate, MLP architecture, and using a Gaussian Mixture Model). We present the results in Table 19. BCQ still underperforms – since the only difference between BC and this version of BCQ at test-time is selecting a GMM action sample uniformly at random versus using the Q-function to select one, this indicates that the Q-function is responsible for poor performance.

### I.1.2 CQL

| Dataset | Default | smaller LR | no DBackup | smaller Batch Size | no Lagrange |
|---|---|---|---|---|---|
| Lift (MG) | $\mathbf{64.0 \pm 2.8}$ | $30.0 \pm 13.4$ | $44.0 \pm 15.0$ | $36.7 \pm 9.0$ | $8.0 \pm 7.1$ |
| Lift (PH) | $\mathbf{92.7 \pm 5.0}$ | $21.3 \pm 5.2$ | $\mathbf{90.7 \pm 5.0}$ | $60.7 \pm 36.5$ | $\mathbf{90.7 \pm 3.8}$ |

Table 20: **CQL Hyperparameter Sensitivity.** We find that CQL (1) is highly sensitive to learning, (2) benefits greatly from larger batch sizes, and (3) benefits from the deterministic backup and Lagrange variants with significant improvements on some datasets.

We investigate the effect of various hyperparameters on CQL in Table 20 for the low-dim Lift MG and PH datasets. First, we see that a smaller learning rate (specifically $10\times$ smaller) for the Q and policy networks leads to a decrease in success rate of over 50%, indicating that CQL is highly sensitive to learning rate. Next, we see that a smaller batch size (specifically 100 instead of 1024) leads to a performance decrease of over 30%, indicating the CQL can greatly benefit from higher batch sizes. The results for excluding the Lagrange variant and the deterministic backup also indicate that these components can help, with substantial improvements in performance for the MG dataset yet marginal improvements for the PH dataset.

## J  GMM Policy Details

| Dataset | Default | No Low Noise Eval |
|---|---|---|
| Square (PH, ld) | **84.0** ± **0.0** | **84.0** ± **3.3** |
| Transport (PH, ld) | **71.3** ± **6.6** | 64.7 ± 5.2 |
| Square (MH, ld) | **78.0** ± **4.3** | **79.3** ± **3.4** |
| Transport (MH, ld) | **65.3** ± **7.4** | 54.7 ± 4.1 |
| Square (PH, im) | **82.0** ± **0.0** | **78.0** ± **4.3** |
| Transport (PH, im) | **72.0** ± **4.3** | **71.3** ± **1.9** |
| Square (MH, im) | **76.7** ± **3.4** | 68.0 ± 0.0 |
| Transport (MH, im) | **42.0** ± **1.6** | **43.3** ± **2.5** |

Table 21: **GMM Low Noise Evaluation Trick.** This table shows the effect of sampling from the GMM instead of using the low-noise-eval trick that we used by default.

As in Acme [88], when using Gaussian Mixture Model (GMM) policies during rollouts, we ignore the learned standard deviations of each mode and instead set it to 1e-4. This amounts to sampling one of the GMM modes instead of sampling from the full GMM distribution. A similar trick is often used when learning Gaussian policies, where at test-time, the mean action of the distribution is used instead of sampling from the distribution. In the table above, we present an ablation for not using this "low-noise-evaluation" (LNE) trick. In most cases, the performance decreases slightly. In early experiments (where the minimum learned std for each Gaussian mode was set to 1e-2 instead of 1e-4), this trick made a more substantial difference. We suggest using this trick by default in all experiments involving GMM policies.

# K  Additional Results on Multi-Human Datasets

This section contains additional results on Multi-Human datasets that were excluded from the main text for space reasons. This includes conducting the Observation Space study on Multi-Human datasets (the main text included results on the Single-Human datasets in Fig. 2a and Table 25), and results on the Lift and Transport Multi-Human data subsets (the main text only included results on the low-dim Can and Square subsets in Table 2, and the image subsets in Appendix H).

| Dataset | Default | + EEF Vel | + Joint | - Rand | - Wrist |
|---|---|---|---|---|---|
| Square(MH, ld) | $78.0 \pm 4.3$ | $16.0 \pm 3.3$ | $15.3 \pm 0.9$ | - | - |
| Transport(MH, ld) | $65.3 \pm 7.4$ | $2.0 \pm 0.0$ | $2.0 \pm 0.0$ | - | - |
| Square(MH, im) | $76.7 \pm 3.4$ | $46.7 \pm 2.5$ | $47.3 \pm 4.1$ | $29.3 \pm 3.8$ | $59.3 \pm 3.4$ |
| Transport(MH, im) | $42.0 \pm 1.6$ | $10.0 \pm 3.3$ | $16.7 \pm 0.9$ | $18.0 \pm 1.6$ | $28.7 \pm 6.8$ |

Table 22: **Observation Space Study (Multi-Human).** This table presents the same observation space study as conducted in Fig 2a, but on the multi-human datasets instead of the proficient-human datasets. The results and conclusions are consistent.

| Dataset | BC | BC-RNN | BCQ | CQL | HBC | IRIS |
|---|---|---|---|---|---|---|
| Lift-Worse | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $97.3 \pm 0.9$ | $13.3 \pm 9.0$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ |
| Lift-Okay | $96.0 \pm 1.6$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $67.3 \pm 10.5$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ |
| Lift-Better | $98.7 \pm 1.9$ | $100.0 \pm 0.0$ | $98.0 \pm 1.6$ | $88.0 \pm 5.9$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ |
| Lift-Worse-Okay | $98.7 \pm 1.9$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $64.7 \pm 2.5$ | $99.3 \pm 0.9$ | $100.0 \pm 0.0$ |
| Lift-Worse-Better | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $98.7 \pm 0.9$ | $75.3 \pm 25.6$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ |
| Lift-Okay-Better | $99.3 \pm 0.9$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | $86.0 \pm 6.5$ | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ |
| Transport-Worse-Worse | $0.6 \pm 0.9$ | $4.7 \pm 0.9$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $4.0 \pm 1.6$ | $6.0 \pm 0.0$ |
| Transport-Okay-Okay | $0.7 \pm 0.9$ | $6.7 \pm 0.9$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $7.3 \pm 1.9$ | $7.7 \pm 1.9$ |
| Transport-Better-Better | $3.3 \pm 0.9$ | $18.7 \pm 3.8$ | $2.0 \pm 0.0$ | $0.0 \pm 0.0$ | $24.0 \pm 3.3$ | $22.0 \pm 3.3$ |
| Transport-Worse-Okay | $1.3 \pm 0.9$ | $5.3 \pm 0.9$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $4.0 \pm 0.0$ | $3.3 \pm 0.9$ |
| Transport-Worse-Better | $7.3 \pm 0.9$ | $22.7 \pm 2.5$ | $0.7 \pm 0.9$ | $0.0 \pm 0.0$ | $35.3 \pm 2.5$ | $25.3 \pm 1.9$ |
| Transport-Okay-Better | $2.7 \pm 0.9$ | $7.3 \pm 2.5$ | $0.0 \pm 0.0$ | $0.0 \pm 0.0$ | $10.0 \pm 1.6$ | $11.3 \pm 3.4$ |

Table 23: **Results on Suboptimal Lift and Transport Human Data Subsets.** We present success rates averaged over 3 seeds for each method across different subsets of the Multi-Human datasets, corresponding to mixtures of demonstrations from "Better", "Adequate", and "Worse" human operators.

| Dataset | BC | BC-RNN | BCQ | CQL |
|---|---|---|---|---|
| Lift-Worse | $98.0 \pm 0.0$ | $100.0 \pm 0.0$ | - | - |
| Lift-Okay | $97.3 \pm 0.9$ | $100.0 \pm 0.0$ | - | - |
| Lift-Better | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | - | - |
| Lift-Worse-Okay | $99.3 \pm 0.9$ | $100.0 \pm 0.0$ | - | - |
| Lift-Worse-Better | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | - | - |
| Lift-Okay-Better | $100.0 \pm 0.0$ | $100.0 \pm 0.0$ | - | - |
| Transport-Worse-Worse | $3.3 \pm 0.9$ | $4.0 \pm 0.0$ | - | - |
| Transport-Okay-Okay | $8.7 \pm 0.9$ | $6.7 \pm 0.9$ | - | - |
| Transport-Better-Better | $32.0 \pm 3.7$ | $39.3 \pm 5.0$ | - | - |
| Transport-Worse-Okay | $5.3 \pm 0.9$ | $4.0 \pm 0.0$ | - | - |
| Transport-Worse-Better | $21.3 \pm 4.1$ | $30.7 \pm 13.6$ | - | - |
| Transport-Okay-Better | $4.7 \pm 2.5$ | $8.7 \pm 2.5$ | - | - |

Table 24: **Results on Suboptimal Lift and Transport Human Data Subsets (Image).** We present success rates averaged over 3 seeds for each method across different subsets of the Multi-Human datasets, corresponding to mixtures of demonstrations from "Better", "Adequate", and "Worse" human operators.

## L  Full Tables

This section contains more detailed tables that were excluded in the main text. These tables correspond to the results in Fig 2, Fig. 3, and Fig 4a.

| Dataset | Default | + EEF Vel | + Joint | - Rand | - Wrist |
|---|---|---|---|---|---|
| Square (ld) | 84.0 ± 0.0 | 42.7 ± 3.4 | 39.3 ± 2.5 | - | - |
| Transport (ld) | 71.3 ± 6.6 | 8.7 ± 0.9 | 10.0 ± 3.3 | - | - |
| Square (im) | 82.0 ± 0.0 | 64.7 ± 0.9 | 58.0 ± 11.4 | 43.3 ± 5.0 | 74.7 ± 3.8 |
| Transport (im) | 72.0 ± 4.3 | 64.7 ± 3.8 | 70.7 ± 2.5 | 46.7 ± 0.9 | 41.3 ± 7.5 |

Table 25: **Observation Space Study.** This table corresponds to the results presented in Fig 2a.

| Dataset | Default | larger LR | no GMM | larger MLP | Shallow Conv | smaller RNN dim |
|---|---|---|---|---|---|---|
| Square (PH, ld) | 84.0 ± 0.0 | 86.0 ± 2.8 | 82.0 ± 0.0 | 82.0 ± 0.0 - | | 81.3 ± 0.9 |
| Transport (PH, ld) | 71.3 ± 6.6 | 64.0 ± 5.9 | 69.3 ± 3.4 | 58.7 ± 6.8 | - | 47.3 ± 2.5 |
| Square (MH, ld) | 78.0 ± 4.3 | 76.7 ± 2.5 | 58.0 ± 1.6 | 73.3 ± 3.4 | - | 58.7 ± 7.4 |
| Transport (MH, ld) | 65.3 ± 7.4 | 49.3 ± 2.5 | 27.3 ± 10.9 | 46.0 ± 3.3 | - | 27.3 ± 12.4 |
| Square (PH, im) | 82.0 ± 0.0 | 41.3 ± 7.7 | 84.0 ± 3.3 | - | 50.0 ± 2.8 | 74.0 ± 3.3 |
| Transport (PH, im) | 72.0 ± 4.3 | 46.7 ± 20.4 | 74.0 ± 4.3 | - | 54.0 ± 3.2 | 65.3 ± 5.2 |
| Square (MH, im) | 76.7 ± 3.4 | 28.7 ± 4.1 | 61.3 ± 0.9 | - | 48.0 ± 3.3 | 58.0 ± 4.3 |
| Transport (MH, im) | 42.0 ± 1.6 | 23.3 ± 4.1 | 41.0 ± 0.3 | - | 16.0 ± 0.0 | 34.0 ± 0.0 |

Table 26: **BC-RNN Hyperparameter Sensitivity.** This table corresponds to the results presented in Fig 2b and Fig 2c.

| Dataset | 20% | 50% | 100% |
|---|---|---|---|
| Lift (ld) | 96.7 ± 2.5 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| Can (ld) | 76.7 ± 5.2 | 97.3 ± 0.9 | 100.0 ± 0.0 |
| Square (ld) | 38.7 ± 6.2 | 67.3 ± 7.7 | 84.0 ± 0.0 |
| Transport (ld) | 6.7 ± 0.9 | 44.0 ± 5.9 | 71.3 ± 6.6 |
| Lift (im) | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| Can (im) | 83.3 ± 1.9 | 97.3 ± 0.9 | 98.0 ± 0.9 |
| Square (im) | 29.3 ± 4.1 | 64.7 ± 4.1 | 82.0 ± 0.0 |
| Transport (im) | 30.7 ± 4.1 | 60.1 ± 4.1 | 72.0 ± 4.3 |

Table 27: **Proficient-Human Dataset Size Ablation.** This table corresponds to the results presented in Fig 3.

| Dataset | 20% | 50% | 100% |
|---|---|---|---|
| Lift (ld) | 100.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| Can (ld) | 79.3 ± 5.0 | 97.3 ± 0.9 | 100.0 ± 0.0 |
| Square (ld) | 32.0 ± 3.3 | 60.7 ± 0.9 | 78.0 ± 4.3 |
| Transport (ld) | 7.3 ± 2.5 | 33.3 ± 7.5 | 65.3 ± 7.4 |
| Lift (im) | 98.0 ± 0.0 | 100.0 ± 0.0 | 100.0 ± 0.0 |
| Can (im) | 77.3 ± 2.5 | 87.3 ± 1.9 | 96.0 ± 1.6 |
| Square (im) | 27.3 ± 1.9 | 50.7 ± 3.8 | 76.7 ± 3.4 |
| Transport (im) | 8.0 ± 3.3 | 25.3 ± 3.8 | 42.0 ± 1.6 |

Table 28: **Multi-Human Dataset Size Ablation.** This table corresponds to the results presented in Fig 3.

| Dataset | Valid (BC) | Last (BC) | Max (BC) | Valid (BC-RNN) | Last (BC-RNN) | Max (BC-RNN) |
|---|---|---|---|---|---|---|
| Square (PH, ld) | 20.0±10.2 | 65.3±0.9 | **78.7±1.9** | 7.3±5.0 | 74.0±7.5 | **84.0±0.0** |
| Transport (PH, ld) | 0.0±0.0 | 11.3±3.8 | **17.3±2.5** | 4.0±5.7 | 59.3±5.0 | **71.3±6.6** |
| Square (PH, im) | 20.6±14.0 | 44.7±4.1 | **62.0±4.9** | 35.3±10.0 | 64.7±9.0 | **82.0±0.0** |
| Transport (PH, im) | 16.0±12.3 | 38.7±11.5 | **55.3±6.2** | 0.0±0.0 | 58.7±5.7 | **72.0±4.3** |

Table 29: **Effect of Policy Selection Criteria.** We compare how performance decreases when choosing the policy to evaluate by using the lowest validation loss, or when using the final trained checkpoint, compared to the best performing policy. Corresponds to results in Fig 4a.