APPENDIX

OVERVIEW

In the appendices below, we provide additional details on the implementation of RoboCrowd, our experiments, and our crowdsourced dataset. We provide a brief overview of each appendix below. For videos, please see our website: https://robocrowd.github.io

Appendix I – *Task Details*

We give descriptions of each of our 6 tasks, as well as renderings and images depicting sample expert demonstrations for each task.

Appendix II – *Dataset Examples*

We provide sample trajectories from our collected dataset including their task and quality annotations, to qualitatively illustrate the diversity of the behaviors in the dataset.

Appendix III – *Additional Dataset Analysis*

We provide further data analysis, including an offline user study to justify our scene choices, additional data quality analysis, and results on users' self-reported Likert ratings of their interactions with the system.

Appendix IV – *Additional Details on Policy Learning Experiments*

We provide additional details on the training and evaluation procedures for our policy learning experiments, as well as further qualitative analysis of the results.

Appendix V – *Additional Details on Software Implementation and Data Annotation*

We provide further details on the graphical user interface, interactive tutorial, software implementation, and data annotation pipeline.

Appendix VI – *Additional Details on Pilot Studies and System Development*

We provide more details on how we designed and refined the system through pilot studies.

Appendix VII – *Overview of Action Chunking with Transfomers (ACT) [1]*

We provide additional background on the Action Chunking with Transfomers (ACT) algorithm.

APPENDIX I
TASK DETAILS

In Tables III to VIII below, we provide a verbal description of the behavior that the expert demonstrations perform for each task. We additionally include a virtual rendering of different segments of a sample demonstration (where the gripper is rendered with increasing opacity for later timesteps). Additionally, we show a timelapse of the overhead camera image observation for the same sample expert demonstration.

APPENDIX II
DATASET EXAMPLES

In Figs. 6 to 8, we give 3 qualitative examples of interaction episodes in our crowdsourced dataset. We illustrate a timelapse of each episode with the overhead camera observation. We also include the task and quality annotations at each timestep, with a verbal description of the episode in the caption.

APPENDIX III
ADDITIONAL DATASET ANALYSIS

In this section, we provide additional data analysis. In Appendix A, we describe an offline study over user preferences for different candies, informing our different scene setups. In Appendix B and Appendix B.1, we examine additional metrics (i.e., tutorial quality and Likert ratings) that correlate with quality of user interaction episodes, and in Appendix C, we provide additional statistics on usage and retention.

A. Justification for Scene Choices

To justify our scene setup and task pairings, we perform an offline survey on user preferences for various candies. On a sample of $N = 16$ users, we find that 81% prefer a Hi-Chew to a Tootsie Roll. Thus, `BinScene` (which includes the `hi-chew` and `tootsie-roll` tasks) allows us investigate whether this preference for material reward shapes task choice when teleoperating demonstrations, when the task is otherwise equivalent besides the material reward. Users exhibit a more mild preference for a Hershey Kiss compared to a small handful of Jelly Beans (with 62% of respondents preferring the Hershey Kiss). `Bin+ZiplocScene` (which includes the `hi-chew-bin` and `hi-chew-ziploc` tasks) allows us to investigate how intrinsic motivation and task difficulty affects user behavior when teleoperating in the case that the material reward (a Hi-Chew) is held constant between the the simpler task and the more challenging task. `Bin+DispenserScene` allows us to investigate this question when the material rewards are different, and users do not exhibit an overall preference for the reward from the harder task (and even mildly prefer the reward from the easier task).

B. Additional Metrics on Demonstration Quality

Our crowdsourced dataset contains rich interaction data per user ID—during and after the interactive tutorial period. This dataset can help to yield insights about which users give higher quality trajectories, and what factors can help predict this quality. As an example, we examine how the quality of interactions *after* the tutorial (i.e., when the user selects tasks in the scene to perform) correlates with quality *during* the tutorial period (i.e., when the user is instructed to complete simple onboarding tasks). Specifically, we examine the distribution of mean quality during task interactions versus minimum quality during the tutorial period; the user's tutorial period is classified as 0 if there is any off-task behavior, 1 if the tutorial is performed but with retrying, and 2 if the tutorial is performed smoothly. We observe a loose positive correlation between higher minimum tutorial quality and mean task quality; and notably, users who produce consistently high quality task demonstrations (quality 3) are more present in the group with high quality tutorials. The tutorial period can therefore be a first-cut proxy at filtering demonstrators by quality.

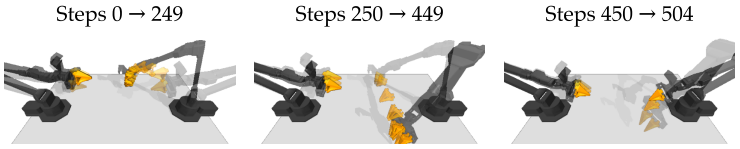
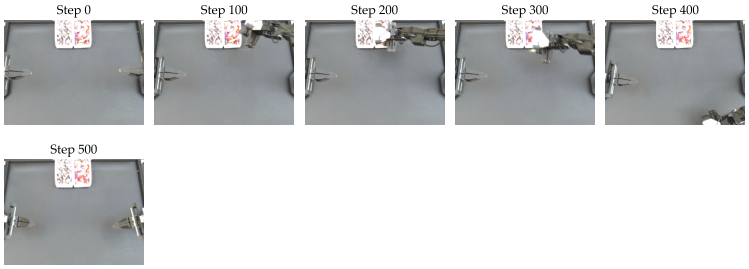
Task Name	Pick up a Hi-Chew (`hi-chew`)
Task Description	Move the right arm towards the candy bin. Grasp one Hi-Chew. Drop it in the End Zone. Finally, return to the home position.
Expert Trajectory Rendering	
Expert Trajectory Timelapse	

TABLE III: Description of the `hi-chew` task, as well as a rendering and timelapse of a sample expert trajectory.

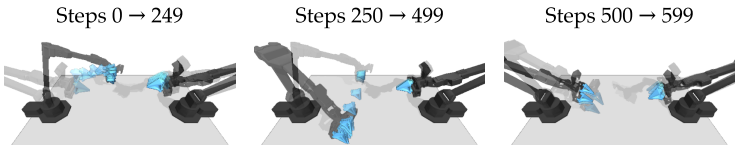
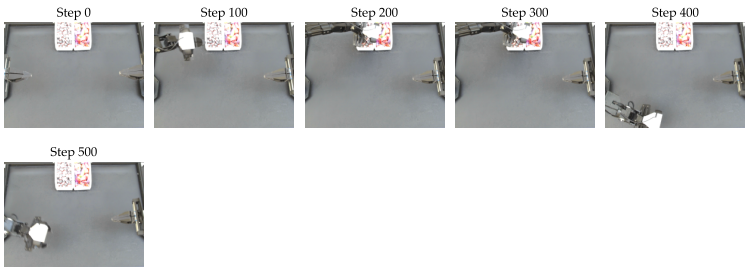
Task Name	Pick up a Tootsie Roll (`tootsie-roll`)
Task Description	Move the left arm towards the candy bin. Grasp one Tootsie Roll. Drop it in the End Zone. Finally, return to the home position.
Expert Trajectory Rendering	
Expert Trajectory Timelapse	

TABLE IV: Description of the `tootsie-roll` task, as well as a rendering and timelapse of a sample expert trajectory.

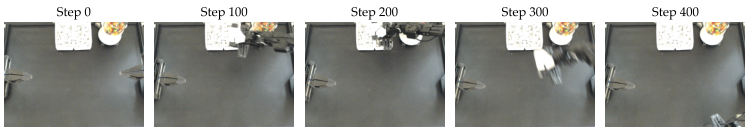
Task Name	Pick up a Hershey Kiss (`hershey-kiss`)
Task Description	Move the right arm **or** the left arm towards the candy bin. Grasp one Hershey Kiss. Drop it in the End Zone. Finally, return to the home position.
Expert Trajectory Rendering	
Expert Trajectory Timelapse	

TABLE V: Description of the `hershey-kiss` task, as well as a rendering and timelapse of a sample expert trajectory.

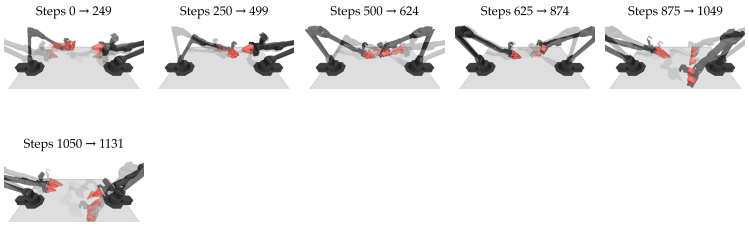
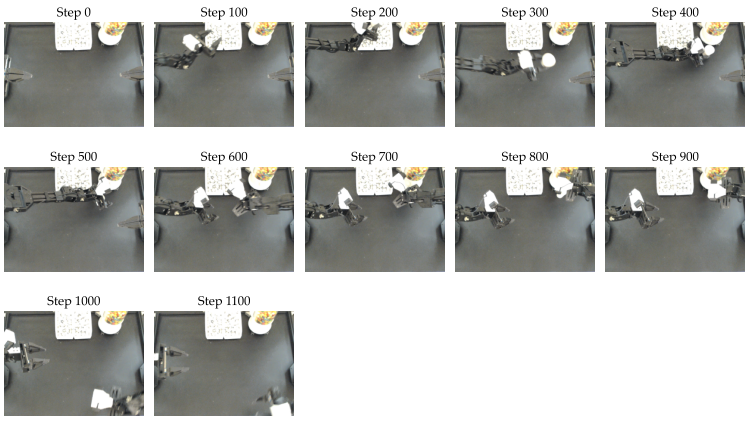
Task Name	Eject a Jelly Bean from the Candy Dispenser (`jelly-bean`)
Task Description	Use the left arm to pull a cup from the cup dispenser. Bring the cup near the lever of the candy dispenser. Use the right arm to align the cup under the lever, then press the lever. Then, use the right arm to pick up the cup and bring it to the End Zone. Finally, return to the home position.
Expert Trajectory Rendering	
Expert Trajectory Timelapse	

TABLE VI: Description of the `jelly-bean` task, as well as a rendering and timelapse of a sample expert trajectory.

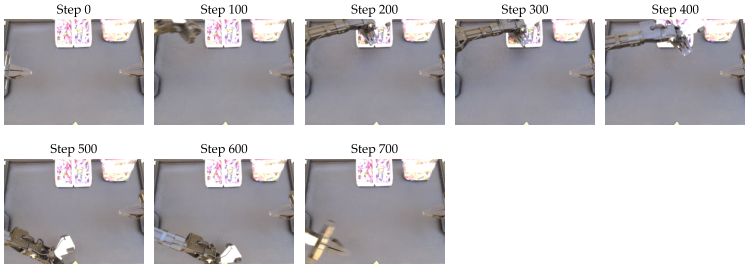
Task Name	Pick up a Hi-Chew from the Bin (`hi-chew-bin`)
Task Description	Move the right arm **or** the left arm towards the candy bin. Grasp one Hi-Chew. Drop it in the End Zone. Finally, return to the home position.
Expert Trajectory Rendering	
Expert Trajectory Timelapse	

TABLE VII: Description of the `hi-chew-bin` task, as well as a rendering and timelapse of a sample expert trajectory.

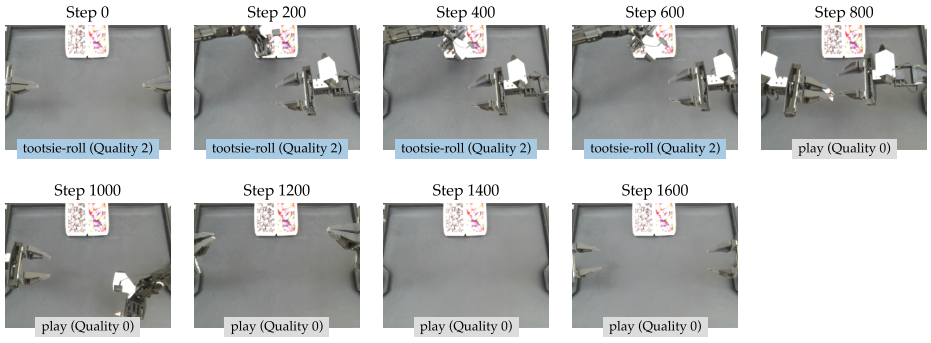


Fig. 6: In this trajectory, the user begins by performing the `tootsie-roll` task with moderate quality—i.e., there are about 3 attempts to grasp the candy, and there is some extraneous movement in the right arm, but the user is otherwise successful at grasping the candy. Before bringing the candy all the way to the End Zone, the user attempts to unwrap the candy. They then hand it over to the other arm, place it in the End Zone, and then move the arms upward. The first half of the episode is marked as `tootsie-roll` (Quality 2) and the latter half of the episode is marked as `play` (Quality 0).

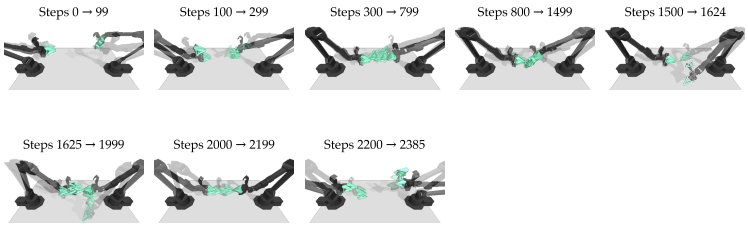
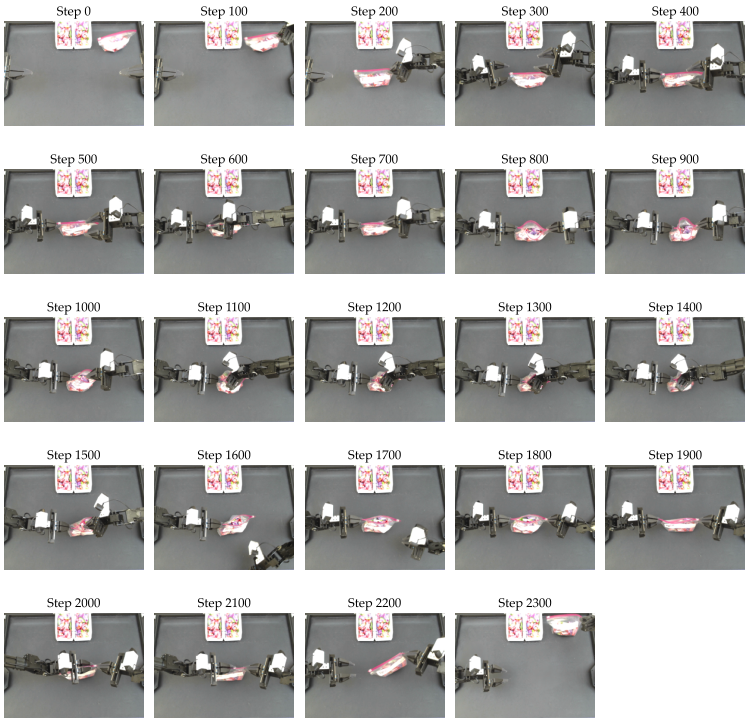
Task Name	Open the Ziploc, Pick up a Hi-Chew, then Close the Ziploc (`hi-chew-ziploc`)
Task Description	Use the right arm to bring the Ziploc bag to the center of the table. Then, use the left arm to hold the Ziploc while pulling the Ziploc tab with the right arm to open the bag. Then, spread the Ziploc open and pick out a Hi-Chew with the right arm, and bring it to the End Zone. Then, use the right arm to hold the Ziploc while pulling the Ziploc tab closed with the left arm. Finally, use the right arm to place the Ziploc back in the corner of the table, and return the arms to the home position.
Expert Trajectory Rendering	
Expert Trajectory Timelapse	

TABLE VIII: Description of the `hi-chew-ziploc` task, as well as a rendering and timelapse of a sample expert trajectory.

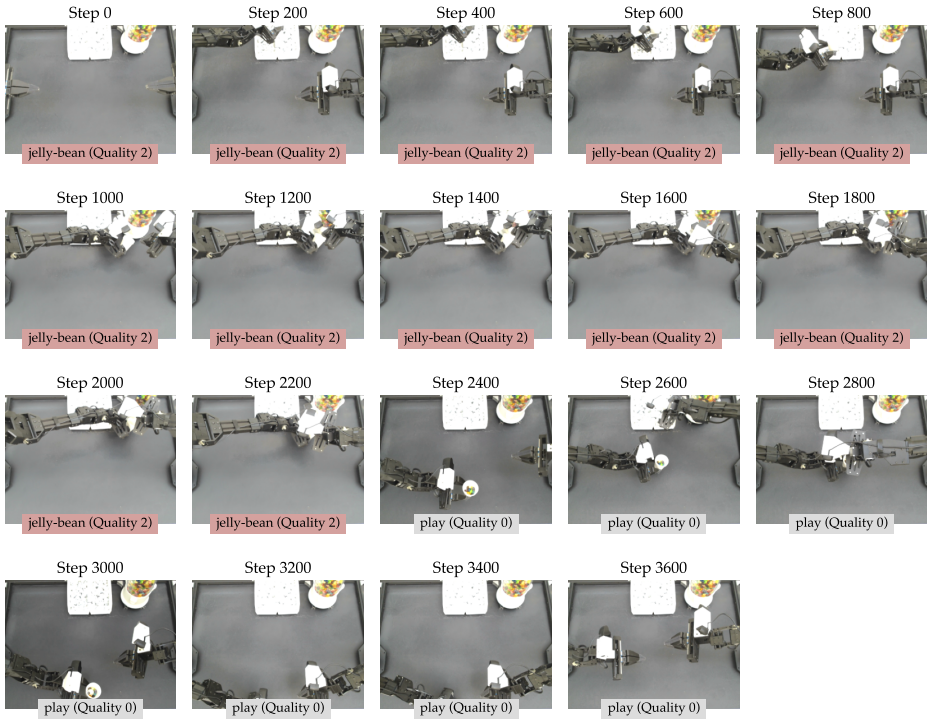


Fig. 7: In this trajectory, the user grasps a cup from the cup dispenser and places it under the lever of the candy machine. They are successful in collecting jelly beans in the cup, though the trajectory includes retrying behavior and is not as smooth as an expert trajectory. The user brings the cup halfway to the End Zone, and then begins behaviors that are not part of the task—i.e., placing a Hershey Kiss in the cup before bringing it to the End Zone. The first part of the episode is marked as `jelly-bean` (Quality 2) and the latter part is marked as `play` (Quality 0).

Fig. 8: In this trajectory, the user correctly moves the Ziploc from the corner of the table to the center of the table, and grasps a Hi-Chew from inside the Ziploc which they bring to the End Zone. They are unsuccessful in closing the Ziploc before episode termination. The user is task-directed for the whole episode, however takes longer than better quality trajectories for this task and performs retrying behavior at each subtask. The whole trajectory is marked as `hi-chew-ziploc` (Quality 1).

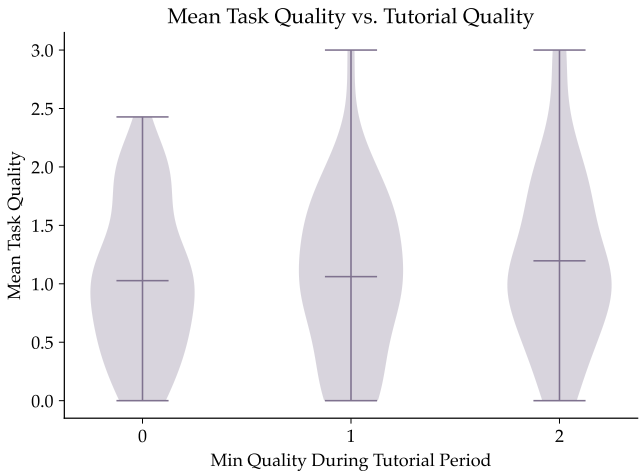


Fig. 9: Distribution of Mean Task Quality versus Minimum Quality during the Tutorial Period.

Learning Rate	1e-5
Batch Size	8
# Encoder Layers	4
# Decoder Layers	7
Feedforward Dimension	3200
Hidden Dimension	512
# Heads	8
Chunk Size	100
KL-weight (β)	10
Dropout	0.1
Backbone	ResNet-18
Image Augmentations	RandomCrop, RandomResize, RandomRotation, ColorJitter

TABLE IX: Hyperparameters for ACT, shared for all experiments.

1) Self-Reported Likert Metrics: After every interaction episode, we prompt the user to answer whether they agree with 3 statements, on a 5-point scale (1 - Strongly Disagree; 2 - Disagree; 3 - Neutral; 4 - Agree; 5 - Strongly Agree).

- `Intuitive`: Controlling the robot was intuitive.
- `Interesting`: Controlling the robot was fun and interesting.
- `Wanted`: The robot accomplished the task in the way that I wanted.

Fig. 10 summarizes the responses to these questions, aggregated by users' minimum ratings to each statement over their interaction episodes. The majority of users agree with all three statements, and most often have the strongest ratings for `Interesting` compared to `Intuitive` and `Wanted`. We find also that there are loose correlations between the manually annotated quality scores for users' interaction episodes and users' self-reported ratings for each of these metrics. Specifically, users who self-report low ratings on each of the three metrics have lower mean quality scores. However, users who self-report high ratings have quality scores that span low to high.

C. Usage and Retention

We illustrate the usage of the RoboCrowd in Fig. 11. We observe significant engagement with RoboCrowd over the two-week collection period: there were $N = 231$ unique users in total. On most days, more than two-thirds of these were new users that had not used the system on prior days. There were a total of 814 interaction episodes distributed throughout the period. The most common time at which users interacted with the system was about 1pm, corresponding to the most trafficked time in the café (lunchtime). We collect 129 interaction episodes in `BinScene` (Day 1), 381 in `Bin+DispenserScene` (Days 2-5), and 307 in `Bin+ZiplocScene` (Days 6-11).

Appendix IV
Additional Details on Policy Learning Experiments

In this section, we give additional details on our policy learning experiments. Appendix A provides training details and hyperparameters, Appendix B provides details on our evaluation procedure, and Appendix C provides additional qualitative discussion of our learned policies.

A. Training Details

For the *Expert* and *Co-train* experiments, we train policies for 200K steps for all tasks. For the *Fine-tune* experiments, we fine-tune the co-trained model (partially trained for 100K steps) for an additional 50K steps on expert data only. We use the implementation of ACT [1] from [54], including the default hyperparameters from [1], as shown in Table IX.

B. Evaluation Details

We perform policy evaluations for 40 trials each, early stopping when policies exhibit excessively jittery or unsafe behavior. While the RoboCrowd training dataset was collected in a café where lighting varies throughout the day, during evaluation, we move the setup to a location with a visually similar background but consistent lighting for controlled evaluations.

For the bin-picking tasks, we define success as the robot arm picking exactly one of the desired candy and bringing it to the End Zone. For our challenging, long-horizon tasks (`jelly-bean` and `hi-chew-ziploc`), success is 0% for all policies, so we instead compare policies via normalized return to measure partial proficiency at tasks. We describe the process for computing normalized return below.

Each of the following subtasks in `jelly-bean` corresponds to 1 point in the episode return: Retrieves Cup from Dispenser; Places Cup Down; Aligns Cup Under Lever; Presses Lever; Collects Jelly Beans in Cup; Picks up Cup; Brings Cup to End Zone. Each of the following subtasks in `hi-chew-ziploc` corresponds to 1 point in the episode return: Picks up Bag; Places Bag in Center of Table; Slides Open; Picks Hi-Chew; Brings Hi-Chew to End Zone; Closes Bag; Places Bag in Corner of Table. For these tasks, we report normalized return—the average return over evaluation trials divided by the maximum return (achieved by all expert demonstrations).

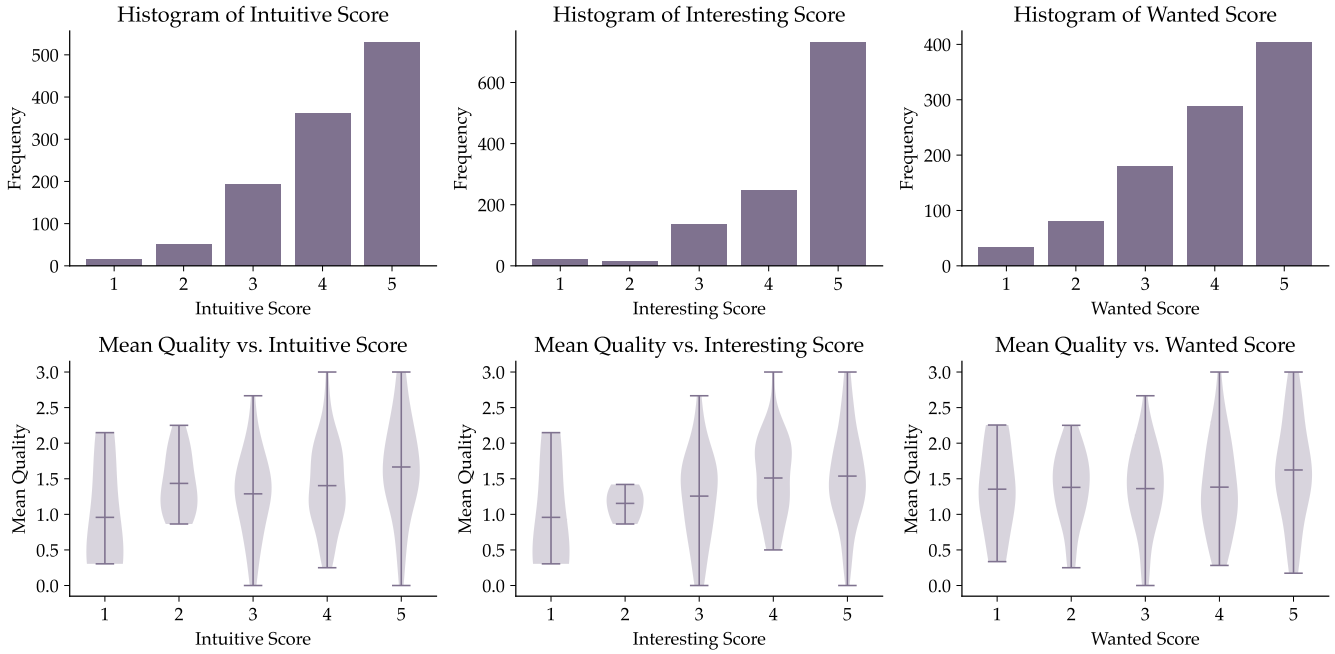


Fig. 10: (*Top*) Histogram of Likert Ratings (aggregated by the user's minimum response over their interaction episodes) for the `Intuitive`, `Interesting`, and `Wanted` questions. (*Bottom*) Distribution of mean quality of interaction episodes for different Likert Ratings for `Intuitive`, `Interesting`, and `Wanted`.

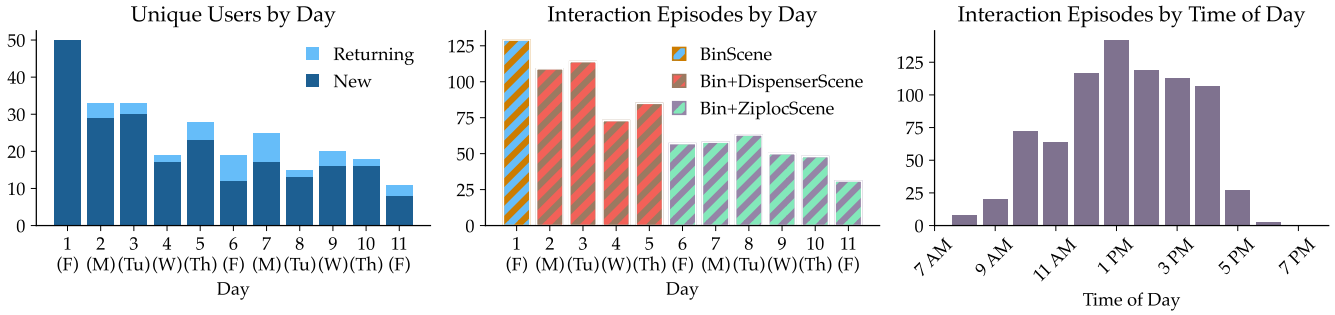


Fig. 11: Statistics on usage over a two-week period: number of users per day (left), number of interaction episodes per day (middle), and distribution of interaction episodes by time of day (right).

C. *Qualitative Analysis of Learned Policies*

We find that in most cases, *Co-train* and/or *Fine-tune* improve upon *Expert*. However, the specific effects vary by task. For example, we find that for the `hi-chew` task, the co-trained policy performs worse than the expert policy, but the fine-tuned policy performs better; whereas with the `hershey-kiss` task, both the co-trained policy and fine-tuned policy perform better. We hypothesize that the crowdsourced data is more useful for `hershey-kiss` because (a) `hershey-kiss` is a more complex task (in that it is more multimodal, i.e., either arm can be used to pick up a Hershey Kiss, and the grasping required needs to be more precise to not crush the Hershey Kiss) and (b) a greater proportion of the `hershey-kiss` data is of higher quality. We notice that the crowdsourced data for `jelly-bean` is especially diverse, and naïvely co-training or fine-tuning underperforms using the expert data only.

Qualitatively, we observe in several cases that the co-trained and fine-tune policies exhibit meaningful but suboptimal behaviors from the crowdsourced data (e.g., picking up multiple objects from the bin instead of one). On the other hand, there are also helpful behaviors from the crowdsourced data (*not* represented in the expert data) that benefit trained policies—e.g., regrasping behavior.

Overall, the RoboCrowd dataset is very diverse, and contains both task-relevant behaviors (of various levels of quality) and free-play behavior. Future work on more sophisticated policy learning methods that leverage these diverse characteristics can help to get the maximum utility out of crowdsourced demonstration data.

APPENDIX V
ADDITIONAL DETAILS ON SOFTWARE IMPLEMENTATION AND DATA ANNOTATION

In this section, we provide additional details on our software interface and implementation, as well as our data annotation pipeline. Appendix A provides an overview of the application flow and interface, Appendix B details the interactive tutorial procedure, Appendix C provides implementation details, and Appendix D details the data annotation pipeline.

A. Application Flow and User Interface

Fig. 12 gives an overview of the flow through the tablet application, and Table X provides screenshots of the major pages referenced in the flowchart. We additionally highlight the Interactive Tutorial in Fig. 13 and the visual warning for collision detection in Fig. 14. We now briefly describe the application flow. To begin a new session, the user taps their ID card on the card reader, which advances the tablet application to a screen where the user can enter a nickname (if they are a new user). They are then directed to the Main Page, where they complete a consent form and the interactive tutorial. From the Main Page, users can also press a "Start Playing" button which directs them to the Task Page, where they can see videos of tasks available in the scene, and can tap on a task to see more details and begin demonstrating the task. For safety, the user receives an audial and visual warning (Fig. 14) if the arms are near-collision. When users are done with the task (i.e., they click a Stop button on the Task Detail Page or they rest the grippers on the mechanical stop), they are asked to mark their demonstration as a success or failure, and fill out a brief survey. The success/failure markings are used as the basis for the points which are added to the user's point total in the Leaderboard, which is accessible from the Main Page; in our experiments, users receive 10 points for successful "easy" tasks (bin-picking) and 20 points for successful "difficult" tasks (the remaining tasks). From the Main Page, users can also choose to provide feedback, or press a Request Help button which immediately notifies the study team (e.g., if the user needs assistance or if the setup requires maintenance).

B. Interactive Tutorial

We provide a zoomed-in version of the pages in the Interactive Tutorial in Fig. 13. The aim of the tutorial is to guide the user on how to start and stop interaction episodes as well as how to puppeteer with ALOHA. Specifically, users are first instructed to wait until ALOHA's arms rise to the home position, and then they are given instructions on how to start puppeteering (by squeezing both sets of grippers on the leader arms). After they do so, the tutorial automatically proceeds to the next stage, where users then are told to gently touch the left and right arms to the table; the goal is to help users get calibrated to the robot's range of motion and degrees of freedom, as well as the types of forces they need to apply to move the arms. Finally, users are given instructions on how to stop the interaction episode, by resting the grippers of the leader arms in the grooves of the mechanical stops. When the user does so, the puppet arms are automatically lowered, and the user is presented a brief video on how to navigate the rest of the interface.

C. Implementation Details

The software application is implemented with React (frontend) and Flask (backend), and uses WebSocket connections to communicate between the user client and backend server. We use a SocketIO-ROS bridge to pass messages between the backend server and robot controller. The robot controller operates at 50Hz and is based on [1]. When the robot is being teleoperated, we run a parallel simulation in MuJoCo [52] which is updated at every time step to detect self-collisions.

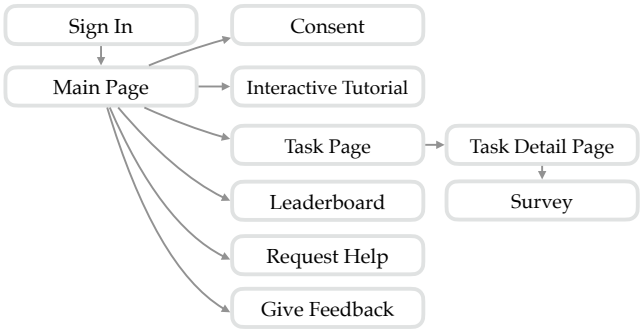


Fig. 12: Flowchart illustration of pages in the user interface.

D. Data Annotation Pipeline

We annotate episodes in our crowdsourced dataset by task and quality. We implement an interface for annotation, which we illustrate in Fig. 15. We annotate episodes by dragging a slider which scrubs through the episode and selecting a task and quality annotation for different segments of the episode. We describe the annotation rules below.

- `play` (Quality 0). All free-play behavior is marked as `play` with quality 0. Play data includes undirected movements and tasks that the user makes up (e.g., trying to unwrap a candy). It also includes extraneous movements before and after the user performs a task.
- `tutorial` (Quality 1–2). Movements associated with the tutorial (e.g., touching the grippers to the table) are marked as Q1 if there is any retrying behavior and Q2 if the motions are smooth.
- `<task>` (Quality 1–3). Task-relevant motions for each of our six tasks are labeled with the task name and a quality from 1 to 3. Q3 is used to describe segments that complete subtasks smoothly with no more than 2 retries. Q2 is used to describe segments that use no more than 4 retries for any one subtask, or that are completed but with slight errors (e.g., grabbing more than 1 candy from a bin). Q1 is used to describe segments that are task-relevant but of poor quality (e.g., more than 4 retries for any one subtask), cause changes to the scene (e.g., dropping a candy on the table), or complete the task in a significantly different manner than the expert demonstrations (e.g., using the opposite arm for any subtask).

Fig. 13: Screenshot of the pages in the interactive tutorial interface.

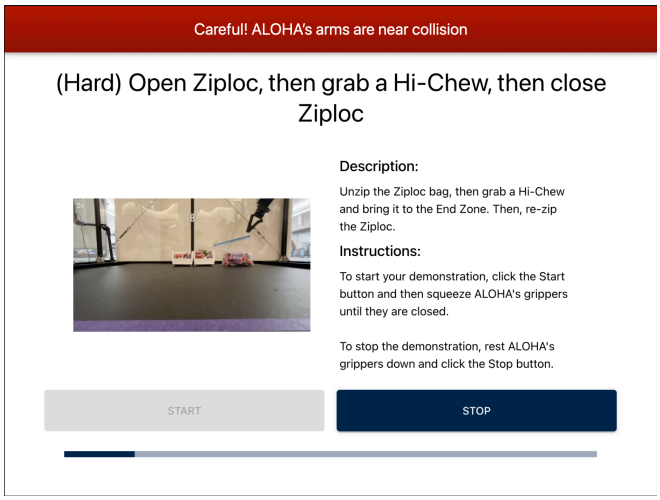


Fig. 14: Screenshot of a visual collision warning on the task page. An audial alarm (beeping sound) is played on the tablet when the visual collision warning appears.

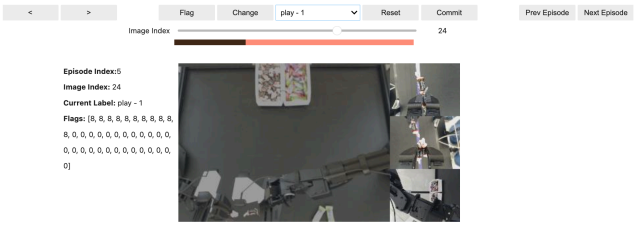


Fig. 15: Screenshot of the data annotation interface. Annotators can scrub through the episode and label segments with task and quality labels, which color codes a bar to visualize the different tasks and qualities in the episode. When the annotator is done labeling an episode, they can "commit" their labels and proceed to the next episode.

Appendix VI
Additional Details on Pilot Studies and System Development

Prior to full system deployment, we conducted pilot studies on a smaller population to help us iterate on our system. We obtained the Institutional Review Board's approval before both the pilot studies and the full deployment. We recruited N=10 participants to interact with the system. In order to mimic organic interactions as closely as possible, we did not provide the participants with any verbal instructions, other than to begin interacting with the system as if they happened upon it organically. Our software interface guided the participants through the consent form and tutorial. Here is a sample of feedback provided by participants, coupled with changes we made to the system.

- *Degrees of Freedom*: Users indicated that puppeteering demonstrations was challenging the first time because they needed to "understand the degrees of freedom" of the robot. To address this feedback, we created a tutorial where the user was guided through how to perform primitive movements of the leader arms (e.g., controlling both puppet arms to touch the bottom of the workspace) before they began interacting with the system.
- *Tutorial Format*: In an initial prototype, our tutorial was a

video that a user would watch before using the system. Users provided feedback that they felt "impatient" and would rather "explore what it is like to interface with the robot" rather than "watch a long video." To address this feedback, we made the tutorial efficient and interactive: 4 steps that the user would perform with the robot after watching them on the screen. The interactive tutorial automatically advances after detecting that each step is complete.

- *Start and Stopping Demonstrations*: In an initial prototype, users begin demonstrations by (1) tapping a Start button on an interface and (2) squeezing the grippers of the leader arms closed. To terminate episodes, they would simply need to (1) leave the arms to rest on the robot body and (2) tap a Stop button on the interface. We received feedback that squeezing the gripper to start episodes "made sense" but the "rest position at the end was confusing." To address this feedback, we designed and 3D printed a mechanical stop for users to rest the arms. We automatically terminate episodes when handles of the leader arms make contact with this mechanical stop.
- *Interface*: In an initial prototype, users would access the interface on their own smartphone by scanning a QR code pasted on the platform. A user reported that they would prefer if more of their interaction would happen "in the position that they will be doing the task." We therefore switched to a tablet interface mounted at the base of the platform, which was accessible when the user sat down to begin interacting with the robot. On the interface itself, users reported that it was "easy to understand."
- *Collisions*: We observed that participants did not actively pay much attention to collisions between the robots, as well as the collision of wrist-camera mounts and objects mounted on the table. To address this, we (1) added collision avoidance between the arms and the table, (2) added an audio-visual alarm when arms were near collision, and (3) mounted objects to the table so that they would not move.

Appendix VII
Overview of Action Chunking with Transformers (ACT)

In this section, we provide a more extended background overview of imitation learning (IL) and the Action Chunking with Transformers (ACT) algorithm [1].

Imitation learning (IL) aims to learn a policy π_θ parameterized by θ given access to a dataset $\mathcal{D}$ composed of expert demonstrations. Defined within the framework of a standard partially observable Markov decision process (POMDP), each trajectory $\xi \in \mathcal{D}$ is a sequence of observation-action transitions $\{(o_0, a_0), \ldots, (o_T, a_T)\}$. Most commonly, IL is instantiated as behavior cloning, which trains π_θ to minimize the negative log-likelihood of data, $\mathcal{L}(\theta) = -\mathbb{E}_{(o,a)\sim\mathcal{D}}[\log \pi_\theta(a|o)]$.

In practice, the human-collected demonstrations in $\mathcal{D}$ may be diverse. To effectively learn from such diverse data, we can condition the policy on a latent variable z, which helps to capture the variability in the demonstrations by representing different modes of behavior. Representing this policy as the decoder in a conditional variational autoencoder (cVAE), we in addition learn an encoder q_ϕ from (observation, action) pairs to the latent space: $q_\phi(z \mid a^t, o^t)$. And we condition our policy on the latent variable: $\pi_\theta(\hat{a}^t \mid o_t, z)$. At test time, we sample latent vectors from the standard normal distribution, $z \sim \mathcal{N}(0,1)$. We regularize the outputs of our encoder towards this distribution via a KL-penalty: $D_{KL}(q_\phi(z \mid a^t, o^t) \,||\, \mathcal{N}(0,1))$. This method is formalized as Action Chunking with Transformers (ACT) [1], an imitation learning algorithm designed to learn from diverse human demonstrations.

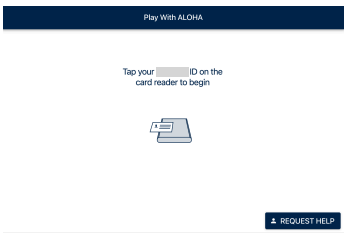
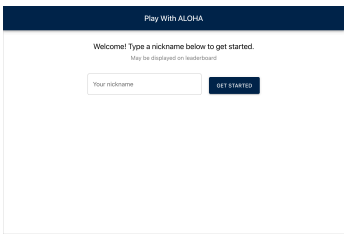
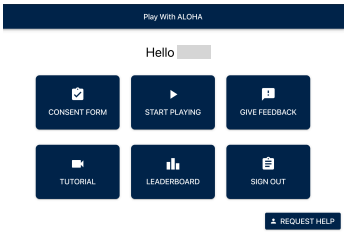
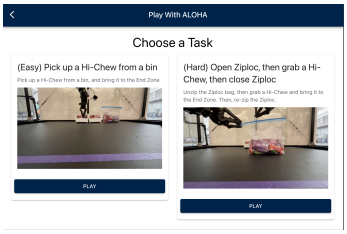
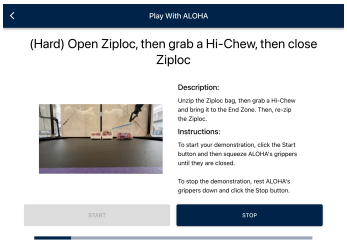
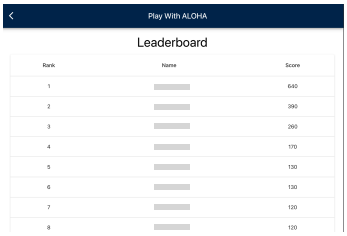
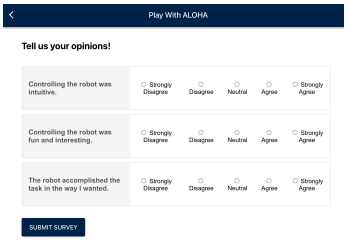
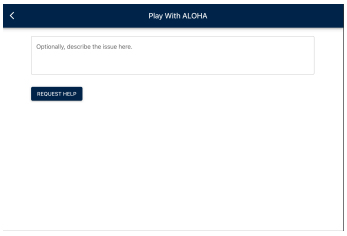
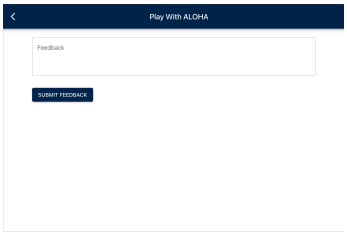
Page Name	Screenshot	Page Name	Screenshot
Sign In (Tap ID Card)		Sign In (Create User Profile)	
Main Page		Interactive Tutorial	
Task Page		Task Detail Page	
Leaderboard		Survey Page	
Request Help		Give Feedback	

TABLE X: Screenshots of pages in the user interface.