Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Robot Learning on the Job: Human-in-the-Loop Autonomy and Learning During Deployment

Huihan Liu[1], Soroush Nasiriany[1], Lance Zhang[1], Zhiyao Bao[1], Yuke Zhu[1]

***Abstract*—With the rapid growth of computing powers and recent advances in deep learning, we have witnessed impressive demonstrations of novel robot capabilities in research settings. Nonetheless, these learning systems exhibit brittle generalization and require excessive training data for practical tasks. To harness the capabilities of state-of-the-art robot learning models while embracing their imperfections, we present Sirius, a principled framework for humans and robots to collaborate through a division of work. In this framework, partially autonomous robots are tasked with handling a major portion of decision-making where they work reliably; meanwhile, human operators monitor the process and intervene in challenging situations. Such a human-robot team ensures safe deployments in complex tasks. Further, we introduce a new learning algorithm to improve the policy's performance on the data collected from the task executions. The core idea is re-weighing training samples with approximated human trust and optimizing the policies with weighted behavioral cloning. We evaluate Sirius in simulation and on real hardware, showing that Sirius consistently outperforms baselines over a collection of contact-rich manipulation tasks, achieving an 8% boost in simulation and 27% on real hardware than the state-of-the-art methods in policy success rate, with twice faster convergence and 85% memory size reduction. Videos and more details are available at https://ut-austin-rpl.github.io/sirius/**

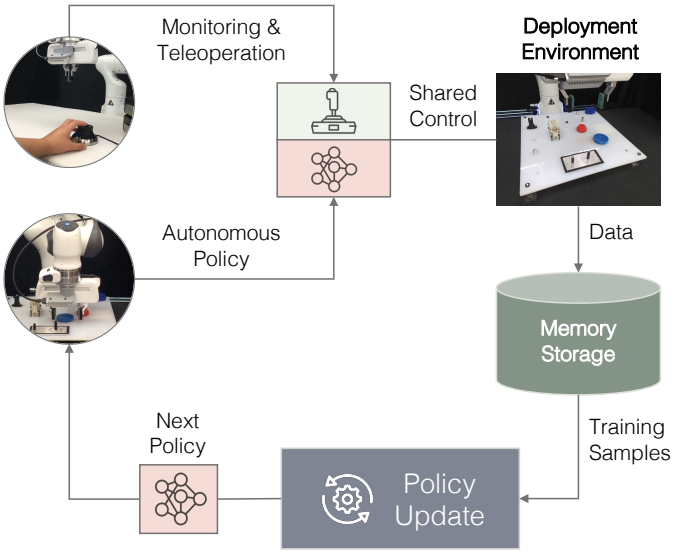

Fig. 1: **Overview of Sirius, our human-in-the-loop learning and deployment framework.** Sirius enables a human and a robot to collaborate on manipulation tasks through shared control. The human monitors the robot's autonomous execution and intervenes to provide corrections through teleoperation. Data from deployments will be used by our algorithm to improve the robot's policy in consecutive rounds of policy learning.

I. Introduction

Recent years have witnessed great strides in deep learning techniques for robotics. In contrast to the traditional form of robot automation, which heavily relies on human engineering, these data-driven approaches show great promise in building robot autonomy that is difficult to design manually. While learning-powered robotics systems have achieved impressive demonstrations in research settings [2, 24, 31], the state-of-the-art robot learning algorithms still fall short of generalization and robustness for widespread deployment in real-world tasks. The dichotomy between rapid research progress and the absence of real-world application stems from the lack of performance guarantees in today's learning systems, especially when using black-box neural networks. It remains opaque to the potential practitioners of these learning systems: how often they fail, in what circumstances the failures occur, and how they can be continually enhanced to address them.

To harness the power of modern robot learning algorithms while embracing their imperfections, a burgeoning body of research has investigated new mechanisms to enable effective human-robot collaborations. Specifically, *shared autonomy* methods [23, 45] aim at combining human input and semi-autonomous robot control to achieve a common task goal. These methods typically use a pre-built robot controller rather than seeking to improve robot autonomy over time. Meanwhile, recent advances in *interactive imitation learning* [6, 25, 37, 46] have aimed to learn policies from human feedback in the learning loop. Although these learning algorithms can improve the overall efficacy of autonomous policies, these policies still fail to meet the performance requirements for real-world deployment.

This work aims at developing a human-in-the-loop learning framework for human-robot collaboration and continual policy learning in deployed environments. We expect our framework to satisfy two key requirements: 1) it ensures task execution to be consistently successful through human-robot teaming, and 2) it allows the learning models to improve continually, such that human workload is reduced as the level of robot autonomy increases. To build such a framework, This idea of *robot learning on the job* resembles the Continuous Integration, Continuous Deployment (CI/CD) principles in software engineering [48]. Realizing this idea for learning-based manipulation invites fundamental challenges.

The foremost challenge is developing the infrastructure for human-robot collaborative manipulation. We develop a system that allows a human operator to monitor and intervene the robot's policy execution (see Fig. 1). The human can take over control when necessary and handle challenging situations to ensure safe and reliable task execution. Meanwhile, human

interventions implicitly reveal the *task structure* and the level of *human trust* in the robot. As recent work [22, 25, 37] indicates, human interventions inform *when* the human lacks trust in the robot, *where* the risk-sensitive task states are, and *how* to traverse these states. We can thus take advantage of the occurrences of human interventions during deployments as informative signals for policy learning.

The subsequent challenge is updating policies on an ever-growing dataset of shifting distributions. As our framework runs over time, the policy would adapt its behaviors through learning, and the human would adjust their intervention patterns accordingly. Deployment data from human-robot teams can be multimodal and suboptimal. Learning from such deployment data requires us to selectively use them for policy updates. We want the robot to learn from good behaviors to reinforce them and also to recover from mistakes and deal with novel situations. At the same time, we want to prevent the robot from copying bad actions that would lead to failure. Our key insight is that we can assess the importance of varying training data based on human interventions for policy learning.

To this end, we develop a simple yet effective learning algorithm that uses the occurrences of human intervention to re-weigh training data. We consider the robot rollouts right before an intervention as "low-quality" (as the human believes the robot is about to fail) and both human demonstrations and interventions as "high-quality" for policy training. We label training samples with different weights and train policies on these samples using weighted behavioral cloning, the state-of-the-art algorithm for imitation learning [47, 56, 63] and offline reinforcement learning [28, 42, 54]. This supervised learning algorithm lends itself to the efficiency and stability of policy optimization on our large-scale and growing dataset.

Furthermore, deploying our system in long-term missions leads to two practical considerations: 1) it incurs a heavy burden of memory storage to store all past experiences over a long duration, and 2) a large number of similar experiences may inundate the small subset of truly valuable data for policy training. We thus examine different memory management strategies, aiming at adaptively adding and removing data samples from the memory storage of fixed size. Our results show that even with 15% of the full memory size, we can retain the same level of performance or achieve even better performance than keeping all data, and moreover enables three times faster convergence for rapid model updates between consecutive rounds.

We name our framework Sirius, the star symbolizing our human-robot team with its binary star system. We evaluate Sirius in two simulated and two real-world tasks requiring contact-rich manipulation with precise motor skills. Compared to the state-of-the-art methods of learning from offline data [28, 39, 42] and interactive imitation learning [37], Sirius achieves higher policy performance and reduced human workload. Sirius reports an 8% boost in policy performance in simulation and 27% on real hardware over the state-of-the-art methods.

II. Related Work

Human-in-the-loop Learning. A human-in-the-loop learning agent utilizes interactive human feedback signals to improve its performance [9, 10, 59]. Human feedback can serve as a rich source of supervision, as humans often have a priori domain information and can interactively guide the agent with respect to its learning progress. Many forms of human feedback exist, such as interventions [25, 37, 50], preferences [3, 8, 32, 53], rankings [4], scalar-valued feedback [35, 55], and human gaze [60]. These feedback forms can be integrated into the learning loop through learning techniques such as policy shaping [19, 27] and reward modeling [11, 33], enabling model updates from asynchronous policy iteration loops [7].

Within the context of robot manipulation, one approach is to incorporate human interventions in imitation learning algorithms [25, 37, 50]. Another approach is to employ deep reinforcement learning algorithms with learned rewards, either from preferences [32, 53] or reward sketching [5]. While these methods have demonstrated higher performance compared to those without humans in the loop, they require a large amount of supervision from humans and also fail to incorporate human control feedback in deployment into the learning loop again to improve model performance. In contrast, we specifically consider the above scenarios which are critical to real-world robotic systems.

Shared Autonomy. Human-robot collaborative control is often necessary for real-world tasks when we do not have full robot autonomy while full human teleoperation control is burdensome. In shared autonomy [13, 18, 23, 45], the control of a system is shared by a human and a robot to accomplish a common goal [52]. The existing literature on shared autonomy focuses on efficient collaborative control from human intent prediction [12, 41, 43]. However, they do not attempt to learn from human intervention feedback, so there is no policy improvement. We examine a context similar to that of shared autonomy where human is involved during the actual deployment of the robot system; however, we also put human control in the feedback loop and use them to improve the learning itself.

Learning from Offline Data. An alternative to the human-in-the-loop paradigm is to learn from fixed robot datasets via imitation learning [14, 36, 44, 61] or offline reinforcement learning (offline RL) [16, 26, 28, 30, 34, 38, 57, 58]. Offline RL algorithms, particularly, have demonstrated promise when trained on large diverse datasets with suboptimal behaviors [1, 29, 49]. Among a number of different methods, advantage-weighed regression methods [28, 42, 54] have recently emerged as a popular approach to offline RL. These methods use a weighted behavior cloning objective to learn the policy, using learned advantage estimates as the weight. In this work, we also use weighted behavior cloning; however, we explicitly leverage human intervention signals from our online human-in-the-loop setting to obtain weights rather than using task rewards to learn advantage-based weights. We show that this leads to superior empirical performance for our

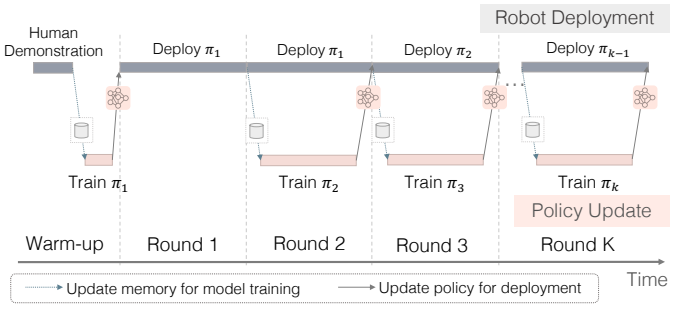

Fig. 2: **Illustration of the workflow in Sirius.** Robot deployment and policy update co-occur in two parallel threads. Deployment data are passed to policy training, while a newly trained policy is deployed to the target environment for task execution.

manipulation tasks.

III. Background and Overview

A. Problem Formulation

We formulate a robot manipulation task as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p_0, \gamma)$ representing the state space, action space, reward function, transition probability, initial state distribution, and discount factor. In this work, we adopt an intervention-based learning framework in which the human can choose to intervene and take control of the robot. Given the current state $s_t \in \mathcal{S}$, the robot action $a_t^R \in \mathcal{A}$ is drawn from the policy $\pi_R(\cdot \mid s_t)$, and the human can override this action with a human action $a_t^H \in \mathcal{A}$. The policy π for the human-robot team can thus be formulated as:

$$\pi(\cdot \mid s_t) = I_H(s_t)\pi_H(\cdot \mid s_t) + (1 - I_H(s_t))\pi_R(\cdot \mid s_t),$$

where I_H is a binary indicator function of human interventions and π_H is the implicit human policy. Our learning objective is two-fold: 1) we want to improve the level of robot autonomy by finding the autonomous policy π_R that maximizes the cumulative rewards $\mathbb{E}_{\pi_R}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_t, a_t, s_{t+1}\right)\right]$, and 2) we want to minimize the human's workload in the system, *i.e.*, the expectation of interventions $\mathbb{E}_{\pi}[I_H(s_t)]$ under the state distribution induced by the team policy π.

B. Weighted Behavioral Cloning Methods

We aim to learn a robot policy π_R with the deployment data to enhance robot autonomy and reduce human costs in human-robot collaboration. Weighted Behavioral Cloning (BC) has recently become one promising approach to learning policies from multimodal and suboptimal data. In standard BC methods, we train a model to mimic the action for each state in the dataset. The objective is to learn a policy π_R parameterized by θ that maximizes the log-likelihood of actions a conditioned on the states s:

$$\theta^* = \arg\max_{\theta} \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} \left[\log \pi_\theta(a \mid s)\right], \tag{1}$$

where (s, a) are samples from the dataset $\mathcal{D}$. For weighted BC, the log-likelihood term of each (s, a) pair is scaled by a weight function $w(s, a)$, which assigns different importance scores to different samples:

$$\theta^* = \arg\max_{\theta} \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} \left[w(s,a) \log \pi_\theta(a \mid s)\right]. \tag{2}$$

The weighted BC framework lays the foundation of several state-of-the-art methods for offline reinforcement learning (RL) [28, 42, 54]. Different weight assignments differentiate high-quality samples from low-quality ones, such that the algorithm prioritizes high-quality samples for learning. In particular, advantage-based offline RL algorithms calculate weights as $w(s, a) = f(Q^\pi(s, a))$, where $f(\cdot)$ is a non-negative scalar function related to the learned advantage estimates $A^\pi(s, a)$. High-advantage samples indicate that their actions likely contribute to higher future returns and, therefore, should be weighted more. Through the sample-weighting scheme, these methods filter out low-advantage samples and focus on learning from the higher-quality ones in the dataset. Nonetheless, effectively learning value estimates can be challenging in practice, especially when the dataset does not cover a sufficiently wide distribution of states and actions—a challenge highlighted by prior work [15, 20]. In the deployment setting, the data only constitute successful trajectories that complete the task eventually. Empirically, we find in Section V that the nature of our deployment data makes today's offline RL methods struggle to learn values.

In contrast to the value learning framework, some prior works [7, 17, 37] have developed weighted BC approaches that are specialized for the human-in-the-loop setting. In particular, Mandlekar et al. [37] proposes Intervention-weighted Regression (IWR) which designs weights based on whether a sample is a human intervention. Inspired by these prior works, we introduce a simple yet practical weighting scheme that harnesses the unique properties of deployment data to learn performant agents. We elaborate on our weighting scheme in the following section.

IV. Sirius: Human-in-the-loop Learning and Deployment

We present Sirius, our human-in-the-loop framework that learns and deploys continually improving policies from human and robot deployment data. First, we define the human-in-the-loop deployment setting and give an overview of our system design. Next, we describe our weighting scheme, which can learn effective policies from mixed, multi-modal data throughout deployment. Finally, we introduce memory management strategies that reduce the computational complexities of policy learning and improve the efficiency of the system.

A. Human-in-the-loop Deployment Framework

Our human-in-the-loop system aims to constantly learn from the deployment experience and human corrective feedback so as to obtain a high-performing robot policy and reduce human workload over time. It consists of two components that happen simultaneously: Robot Deployment and Policy Update. In Robot Deployment (top thread in Fig. 2), the robot performs task executions with human monitoring; in Policy

Update (bottom thread), the system improves the policy with the deployment data for the next round of task execution.

The system starts with an initial policy in the warm-up phase, where we bootstrap a robot policy π_1 trained on a small number of human demonstrations. Initially, the memory buffer comprises a set of human demonstration trajectories $\mathcal{D}^0 = \{\tau_j\}$, where each trajectory $\tau_j = \{s_t, a_t, r_t, c_t = \texttt{demo}\}$ consists of the states, actions, task rewards, and the data class type flag c_t indicating that these trajectories are human demonstrations.

Upon training the initial policy π_1, we deploy the robot to perform the task, and in the process, we collect a set of trajectories to improve the policy. A human operator who continuously monitors the robot's execution will intervene based on whether the robot has performed or will perform suboptimal behaviors. Note that we adapt human-gated control [25] rather than robot-gated control [22] to guarantee task execution success and trustworthiness of the system for real-world deployment. Through this process, we obtain a new dataset $\mathcal{D}'$ of trajectories $\tau_j = \{s_t, a_t, r_t, c_t\}$, where c_t either indicates the transition is a *robot action* ($c_t = \texttt{robot}$) or a *human intervention* ($c_t = \texttt{intv}$). We append this data to the existing memory buffer collected so far $\mathcal{D}^1 \leftarrow \mathcal{D}^0 \cup \mathcal{D}'$, and train a new policy π_2 on this new dataset.

In subsequent rounds, we deploy the robot to collect new data while simultaneously updating the policy. We define "Round" as the interval for policy update and deployment: It consists of the completion of training for one policy, and at the same time, the collection of one set of deployment data. In Round i, we train for policy π_i using all previous data. Meanwhile, the robot is continuously being deployed using the current best policy π_{i-1}, and gathered deployment data $\mathcal{D}'$. At the end of round i we append this data to the existing memory buffer collected so far $\mathcal{D}^i \leftarrow \mathcal{D}^{i-1} \cup \mathcal{D}'$ and train a new policy π_{i+1} on this aggregated dataset.

Our system aggregates data from deployment environments over long-term deployments. This presents a unique set of challenges: first, the generated data comes from **mixed distributions** consisting of robot policy actions, human interventions, and human demonstrations; also, the system produces data that is **constantly growing in size**, imposing memory burden and computational inefficiency for learning algorithms. We address these challenges in the following sections.

B. Human-in-the-loop Policy Learning

We present a simple yet effective learning method that takes advantage of the unique characteristics of deployment data to learn effective policies. We have a critical insight that human interventions provide informative signals of human trust and human judgement of the robot executions, which we will use to guide the design of our algorithm. The core idea of our approach is to harness the structure of the human correction feedback to re-weigh training samples based on an approximate quality score. With these weighted samples, we train the policy with the weighted behavioral cloning method to learn the policy on mixed-quality data. Our approach is

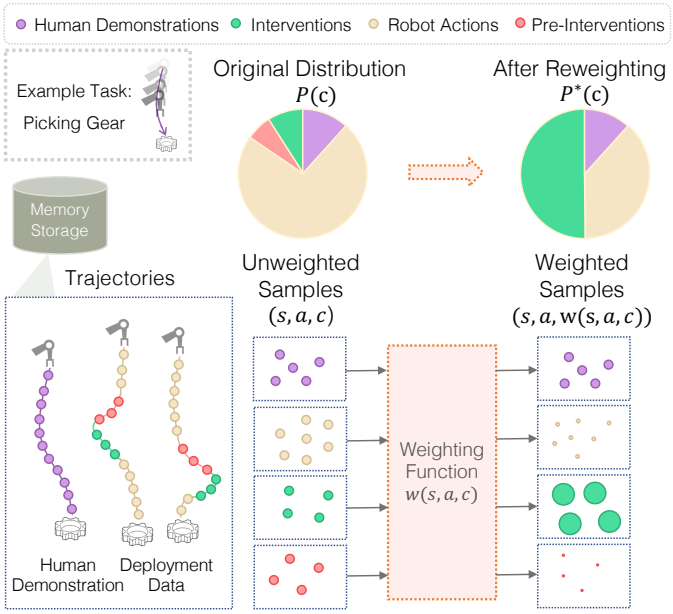

Fig. 3: **Overview of our human-in-the-loop learning model.** We maintain an ever-growing database of diverse experiences spanning four categories: human demonstrations, autonomous robot data, human interventions, and transitions preceding interventions which we call pre-interventions. We set weights according to these four categories, with a high weight given to interventions over other categories. We use these weighted samples to continually learn vision-based manipulation policies during deployment.

motivated by two insights on how the human intervention structure could be used.

Our first intuition is that human intervention samples are highly important samples and should be prioritized in learning. Human-operated samples are expensive to obtain and should be optimized in general, but human intervention occurs in situations where the robot is unable to complete the task and requires help. These are risk-sensitive task states, so data in these regions are highly valuable. Therefore, these state-action pairs should be ranked high by the weighting function, and we should upweight the human intervention samples such that these samples will positively influence learning more.

Moreover, we should not only make use of *what* human samples to use, but also *when* the human samples take place. We make the critical observation that when the robot operates autonomously, it usually performs reasonable behaviors. But when it demands interventions, it is when the robot has made mistakes or has performed suboptimal behaviors. Therefore, human interventions implicitly signify human value judgment of the robot behavior—the samples before human interventions are less desirable and of lower quality. We aim to minimize their impact on learning.

With these insights, we devise a weighting scheme according to intervention-guided data class types. Recall that each sample (s, a, r, c) in our dataset contains a data class type c, indicating whether the sample denotes a human demonstration action, robot action, or human intervention action. To incorporate the timing of human interventions, we distinguish and penalize the samples taken prior to each human intervention. We define the segment preceding each human intervention as a

Algorithm 1 Human-in-the-loop Learning at Deployment

Notations
L: memory buffer maximum fixed size
X: maximum deployment rounds
M: number of initial human demonstration trajectories
K: number of rollout episodes in each deployment round
b: batch size
n: number of gradient steps in each learning round
α: policy learning rate

▷ *warmstart phase*
Collect M human demonstrations $\tau_1, \ldots, \tau_M$
$\mathcal{D}^0 \leftarrow \{\tau_1, \ldots, \tau_M\}$
Initialize BC policy π_1^θ:
$\theta^* = \arg\max_\theta \mathbb{E}_{(s,a)\sim\mathcal{D}^0} \left[\log \pi_1^\theta(a \mid s)\right]$

▷ *initial deployment data*
$\mathcal{D}^1 \leftarrow$ DEPLOYMENT(π_1^θ, $\mathcal{D}^0$)

▷ *deployment-learning loop*
for $i \leftarrow 1$ to X **do**
Run in parallel:
$\mathcal{D}^{i+1} \leftarrow$DEPLOYMENT($\pi_i^\theta$, $\mathcal{D}^i$)
$\pi_{i+1}^\theta \leftarrow$LEARNING($\mathcal{D}^i$)

▷ *deployment thread*
function DEPLOYMENT(π_θ, $\mathcal{D}$)
Collect rollout episodes $\tau_1, \ldots, \tau_K \sim p_{\pi_\theta}(\tau)$
$\mathcal{D}^+ \leftarrow \mathcal{D} \cup \{\tau_1, \ldots, \tau_K\}$
if $|\mathcal{D}^+| > L$ **then**
Discard trajectories in $\mathcal{D}^+$ s.t. $|\mathcal{D}^+| \leq L$
with a memory management strategy (in IV-C)
return $\mathcal{D}^+$

▷ *learning thread*
function LEARNING($\mathcal{D}$)
Initialize π_θ
for each class c **do**
$\mathcal{D}_c \leftarrow \{(s, a, c') \in \mathcal{D} \mid c' = c\}$
$P(c) \leftarrow |\mathcal{D}_c|/|\mathcal{D}|$
Obtain $P^*(c)$ (see IV-D)
for n gradient steps **do**
Sample mini-batch $\left(s^i, a^i, c^i\right)_{i=1}^b \sim \mathcal{D}$
Compute $w(s^i, a^i, c^i) \leftarrow \frac{P^*(c^i)}{P(c^i)}$ for the mini-batch
$\mathcal{L}_\pi(\theta) = -\frac{1}{b}\sum_i \left[w(s^i, a^i, c^i) \cdot \log \pi_\theta(a^i \mid s^i)\right]$
$\theta \leftarrow \theta \ - \ \alpha \nabla_\theta \mathcal{L}_\pi(\theta)$
return π_θ

separate class, pre-intervention (`preintv`) (see Fig. 3). This classification is based on the implicit human evaluation from the human partner, thresholding the robot samples into either normal `robot` samples or suboptimal `preintv` samples. Overall, this yields four class types $c \in \{$`demo`, `intv`, `robot`, `preintv`$\}$.

We derive the weight for each individual sample according to its corresponding class type c. Suppose the dataset $\mathcal{D}$ has total number of samples N, and n_c is the number of samples that is class c. We use $\mathcal{D}_c$ to represent the collection of samples of class c in $\mathcal{D}$. The original class distribution is $P(c) = n_c/N$ for class c, and the unweighted BC objective under this distribution is:

$$\begin{aligned} &\arg\max_\theta \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}} \left[\log \pi_\theta(a \mid s)\right] \\ =\ &\arg\max_\theta \mathop{\mathbb{E}}_{P(c)} \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{D}_c} \left[\log \pi_\theta(a \mid s)\right]. \end{aligned} \tag{3}$$

In a long-term deployment setting, most data will be robot actions, and human interventions usually constitute a small ratio of the dataset samples, since interventions only happen at critical regions in a trajectory; the pre-intervention samples constitute a small but non-negligible proportion which can have detrimental effects (see Fig. 3, left pie chart). We will now change the class distribution to a new distribution $P^*(c)$, in which we increase the ratio of human intervention samples and decrease the ratio of the pre-intervention samples (see Fig. 3, right pie chart). Under this new distribution, the weight $w(s, a, c)$ of the training samples in each individual class c can be equivalently set as $w(s, a, c) = P^*(c)/P(c)$ by the rule of importance sampling. We outline the details of our specific distribution $P^*(c)$ in Sec. IV-D. This way, we obtain the sample weights for weighted BC, leveraging the inherent structure of human-robot team data.

C. Memory Management

As the deployment continues and the dataset increases, large data slows down training convergence and takes up excessive memory space. We hypothesize that forgetting (routinely discarding samples from memory) helps prioritize important and useful experiences for learning, speeding up convergence and even further improving policy. Moreover, the right kind of forgetting matters, since we want to preserve the data that is most beneficial to learning. Therefore, we would like to investigate the following question—with limited data storage and a never-ending deployment data flow, how do we absorb the most useful data and preserve more valuable information for learning?

We assume that we have a fixed-size memory buffer that replaces existing samples with new ones when full. We consider five strategies for managing the memory buffer of deployment data. Each strategy tests out a different hypothesis listed below:

1) **LFI** (Least-Frequently-Intervened): first reject samples from trajectories with the least interventions.
(Preserving the most human intervened trajectories

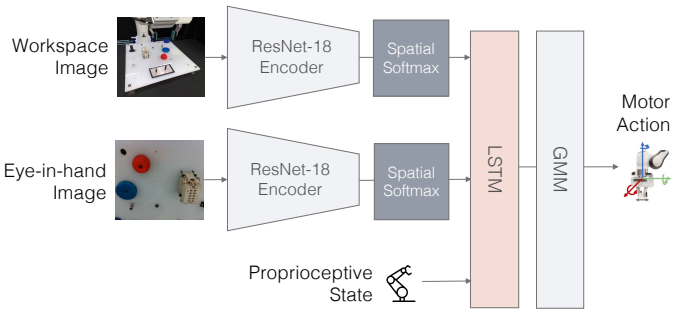

Fig. 4: **Policy Architecture.** Our vision-based policy uses BC-RNN as our policy backbone. Our inputs are workspace camera image and eye-in-hand camera image, as well as robot proprioceptive states.

keeps the most valuable human and critical state examples, which helps learning the most.)

2) **MFI** (Most-Frequently-Intervened): first reject samples from trajectories with the most interventions.
(Successful, unintervened robot trajectories yield higher quality data for learning compared to those that require intervention.)
3) **FIFO** (First-In-First-Out): reject samples in the order that they were added to the buffer.
(More recent data from a higher performing policy are higher quality data for learning.)
4) **FILO** (First-In-Last-Out): reject the most recently added samples first.
(Initial data from a worse performing policy have greater state coverage and data diversity for learning.)
5) **Uniform**: reject samples uniformly at random.
(Uniformly selecting trajectories can yield a balanced mix of diverse samples, aiding in the learning process.)

With the intervention-guided weighting scheme for policy update and memory management strategies, we present the overall workflow of human-in-the-loop learning in deployment in Algorithm 1.

D. Implementation Details

For the robot policy (see Fig. 4), we adopt BC-RNN [39], the state-of-the-art behavioral cloning algorithm, as our model backbone. We use ResNet-18 encoders [21] to encode third person and eye-in-hand images [36, 39]. We concatenate image features with robot proprioceptive state as input to the policy. The network outputs a Gaussian Mixture Model (GMM) distribution over actions.

For our intervention-guided weighting scheme, we set $P^*(\texttt{intv}) = \frac{1}{2}$. The 50% ratio is adapted from prior work [37] that increases the weight of intervention to a reasonable level. We conduct an ablation study in Section V how changing $P^*(\texttt{intv})$ affects the policy performance. We set $P^*(\texttt{preintv}) = 0$, essentially nullifying the impact of pre-intervention samples. The `demo` weight maintains the true ratio of demonstration samples in the dataset: $P^*(\texttt{demo}) = P(\texttt{demo})$. Finally, $P^*(\texttt{robot})$ adjusts itself accordingly. Under this new distribution, we implicitly decrease the proportion of the `robot` class (see Fig. 3) due to increasing the proportion of the `intv` class. Note that the ratio of the demonstration remains unchanged as they are still important and useful samples to learn from, especially during initial rounds of updates when the robot generates lower-quality data. This is in contrast to IWR by Mandlekar et al. [37], which treats all non-intervention samples as a single class, thus lowering the contribution of demonstrations from their unweighted ratio. The weight for each individual sample is $w(s, a, c) = P^*(c)/P(c)$, as discussed in Section IV-B.

We set a segment of length ℓ before each human intervention as the class `preintv`. The optimal choice on the hyperparameter ℓ depends on the *human reaction time*, which quantifies how fast the human operator reacted to the robot's undesired behavior. Prior works [50, 51] indicate that a response delay exists between the time the robot starts to perform mistakes and the time human actually perform corrective interventions. Our empirical observation based on our human operator shows an average reaction time of 2 seconds, roughly corresponding to the time of 15 robot actions. We thus set $\ell = 15$.

V. Experiments

In our experiments, we seek to answer the following research questions: 1) How effective is Sirius in improving autonomous robot policy performance over time? 2) Can this system reduce human workload over time? 3) How do the individual design choices in our learning algorithm affect overall performance? and 4) Which memory management strategy is most effective for learning with constrained memory storage?

A. Tasks

We design a set of simulated and real-world tasks that resemble common industrial tasks in manufacturing and logistics. We consider long-horizon tasks that require precise contact-rich manipulation, necessitating human guidance. For all tasks, we use a Franka Emika Panda robot arm equipped with a parallel jaw gripper. Both the agent and human control the robot in task space. We use a SpaceMouse as the human interface device to intervene.

We systematically evaluate the performance of our method and baselines in the robosuite simulator [62]. We choose the two most challenging contact-rich manipulation tasks in the robomimic benchmark [39]:

Nut Assembly. The robot picks up a square nut from the table and inserts the nut into a column.

Tool Hang. The robot picks up a hook piece and inserts it into a very small hole, then hangs a wrench on the hook. As noted in robomimic [39], this is a difficult task requiring precise and dexterous control.

In the real world, we design two tasks representative of industrial assembly and food packaging applications:

Gear Insertion. The robot picks up two gears on the NIST board and inserts each of them onto the gear shafts.

Coffee Pod Packing. The robot opens a drawer, places a coffee pod into the pod holder, and closes the drawer.

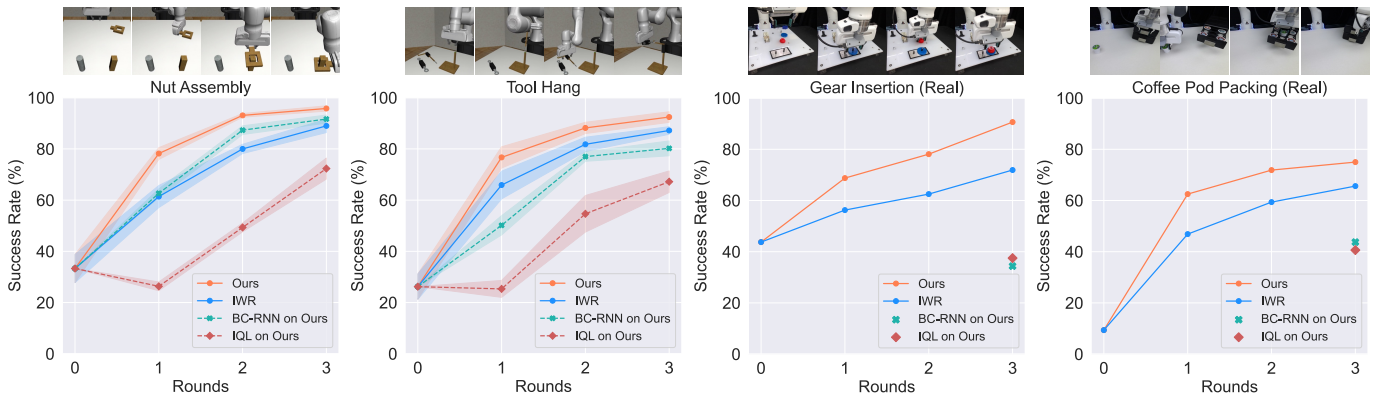

Fig. 5: **Quantitative evaluations.** We compare our method with human-in-the-loop learning, imitation learning, and offline reinforcement learning baselines. Our results in simulated and real-world tasks show steady performance improvements of the autonomous policies over rounds. Our model reports the highest performance in all four tasks after three rounds of deployments and policy updates. Solid line: human-in-the-loop; dashed line: offline learning on data from our method.

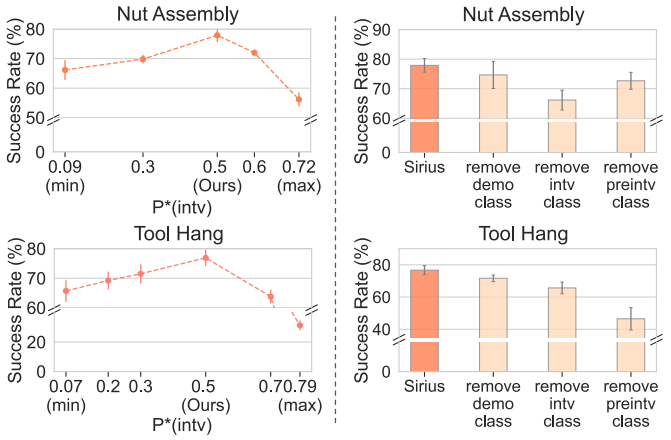

Fig. 6: **(Left) Ablation on intervention ratio weight.** We show how policy performance first increase then decrease as $P^*(\texttt{intv})$ increases, pearking at $P^*(\texttt{intv}) = 0.5$. **(Right) Ablation on weight function design.** Our results show that removing each class label hurts model performance.

B. Baselines and Evaluation Protocol

We compare our method with the state-of-the-art human-in-the-loop learning method for robot manipulation, Intervention Weighted Regression (**IWR**) [37]. Furthermore, to ablate the impacts of algorithms versus data distributions, we compare the state-of-the-art imitation learning algorithm **BC-RNN** [39] and offline RL algorithm Implicit Q-Learning (**IQL**) [28]. We run these two latter baselines on the deployment data generated by our method for a fair comparison.

To mimic the intervention-guided weights for IQL, we use the following rewards after hyperparameter optimization: $r = 1.0$ upon task success, $r = 0.25$ for intervention states, $r = -0.25$ for pre-intervention states, and $r = 0$ for all other states. We also run IQL in a sparse reward setting but find it underperformed. Note that in contrast to our method, IQL requires additional information on task rewards, which may be expensive to obtain in real-world settings.

To provide a fair comparison with existing human-in-the-loop methods, we follow the round update protocol established by prior work [25, 37]: three rounds of policy learning and deployment, where each round deployment runs until the number of intervention samples reaches one third of the initial human demonstration samples.

We benchmark human-in-the-loop deployment systems in two aspects: 1) **Policy Performance.** Our human-robot team achieves a reliable task success of 100%. Here we evaluate the success rate of the autonomous policy after each round of model update; and 2) **Human Workload.** We measure human workload as the percentage of intervention in the trajectories in each round. We perform rigorous evaluations of policy performance as follows:

- *Simulation experiments*: We evaluate the success rate of each method across 3 seeds. For each seed, we evaluate the success rate at a set of regularly spaced training checkpoints and record the average over the top three performing checkpoints to avoid outliers. For each checkpoint, we evaluate whether the agent successfully completed the task over 100 trials.
- *Real-world experiments*: We evaluate each method for one seed due to the high time cost for real robot evaluation. Since real robot evaluations are subject to noise and variation across checkpoints, we first perform an initial evaluation of different checkpoints (5 checkpoints) for each method, evaluating each of them for a small number of trials (5 trials). For the checkpoint that gives the best initial quantitative behavior, we perform 32 trials and report the success rate over them.

C. Experiment Results

Quantitative Results. We show in Fig. 5 that our method significantly outperforms the baselines on our evaluation tasks. Our method consistently outperforms IWR over the rounds. We attribute this difference to our fine-grained weighting scheme, enabling the method to better differentiate high-quality and suboptimal samples. This advantage over IWR cascades across the rounds, as we obtain a better policy, which in turn yields better deployment data.

We also show that our method significantly outperforms the BC-RNN and IQL baselines under the same dataset distribution. This highlights the importance of our weighting scheme

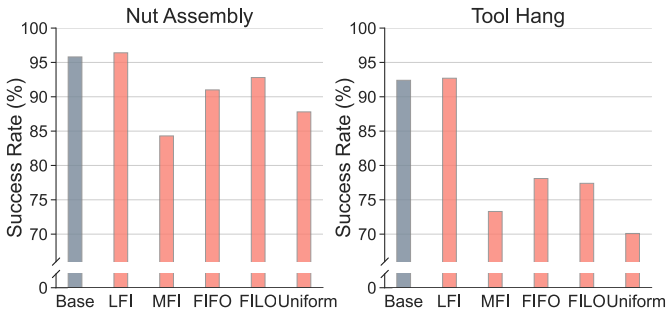

Fig. 7: **Ablation on memory management strategies.** We study the five different strategies introduced in Section IV-C. LFI (discarding least frequently intervened trajectories) matches and even yields better performance over keeping all data samples (Base) while taking much less memory storage.

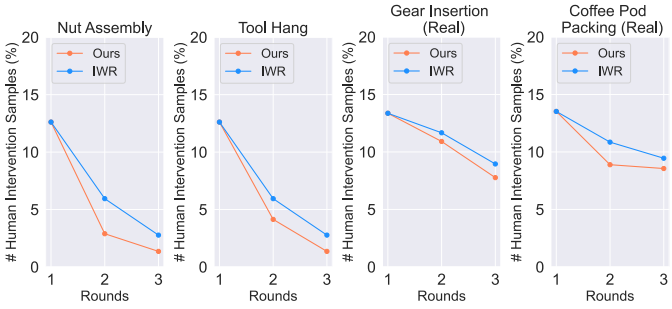

Fig. 8: **Human Intervention Sample Ratio.** We evaluate the human intervention sample ratio for the four tasks. The human intervention sample ratio decreases over deployment round updates. Our methods have a larger reduction in human intervention ratio as compared with IWR.

— BC-RNN performs poorly due to copying the suboptimal behaviors in the dataset, while IQL fails to learn values as weights that yield effective policy performance.

Ablation Studies. We perform an ablation study to examine the contribution of each component in our weighting scheme in Fig. 6 (Right). We study how removing each class, *i.e.*, treating each class as the robot action class (and thus removing the special weight for that class), affects the policy performance:

- **remove demo class:** not preserving the true ratio of `demo` class, which lowers its contribution (see IV-D).
- **remove intv class:** not upweighting the `intv` class, which is equivalent to (min) in Fig. 6 (Left).
- **remove preintv class:** not downweighting the `preintv` class but treating it as `robot` class.

We run each ablated version of our method on Round 1 data for the simulation tasks. We choose Round 1 data for this study because they are generated from the initial BC-RNN policy rather than biased toward data generated from our method. As shown in Fig. 6 (Right), removing any class weight hurts the policy performance. This shows the effectiveness of our fine-grained weighting scheme, where each class contributes differently to the learning of the deployment data.

We also conduct an in-depth study on the influence of human intervention reweighting ratio $P^*(\texttt{intv})$. In the unweighted distribution, the human intervention samples take up a small proportion of the dataset size, which we denote as the *minimum ratio*; the maximum ratio it can take is to nullify the proportion of `robot` samples altogether (so that the dataset only constitutes human demonstrations and human interventions). We run our method with a different ratio ranging from minimum to maximum using Round 1 data on both simulation tasks. The specific range for Nut Assembly and Tool Hang can be found in Fig. 6 (Left). The overall trend is that the policy performance peaks at $P^*(\texttt{intv}) = 0.5$, and is worse when $P^*(\texttt{intv})$ gets larger or smaller. Our intuition is that if the intervention ratio is too small, we are not making the best use of the intervention samples; if it is too large, it will limit the diversity of training data. Either way has an adversarial effect.

Analysis on Memory Management. We compare the effectiveness of Memory Management strategies in Section IV-C at deployment. Fig. 7 shows the result of memory size reduction on the two simulation tasks in Round 3, where the Nut Assembly accumulated 3000+ trajectories and the Tool Hang task 1600+ trajectories. By capping our memory buffer size at 500 trajectories, we manage to reduce memory size to a much small proportion of the original dataset size (15% for Nut Assembly and 30% for Tool Hang).

Among all of the strategies, LFI (discarding least frequently intervened trajectories) is the only strategy that matches and even yields better performance over keeping all data samples (Base). In addition to minimizing storage requirements, LFI also improves learning efficiency. Under LFI, the policy converged twice as fast as Base for both tasks (where we define convergence as the number of epochs to reach 90% success rate). The faster convergence speed, in turn, yields faster model iterations in real-world deployments.

There are a number of potential explanations for the superior performance of LFI. First, note that among all of the strategies, LFI preserves the largest number of human intervention samples. This suggests that human interventions have high intrinsic value to our learning algorithm, as they help to ensure robust policy execution under suboptimal scenarios. Another perspective is that LFI preserves the more frequently intervened trajectories, which exhibit wider state coverage and a diverse array of events. This facilitates the trained policies to operate effectively under rare and unexpected scenarios. MFI (discarding most intervened trajectories) has the opposite effect, favoring trajectories that require less human supervision and often exhibit less diverse behaviors. The results on FIFO and FILO suggest that managing samples according to deployment time is not the most effective strategy, as valuable training data can be collected all throughout the deployment of the system. Finally, the naïve Uniform strategy is ineffective as it does not incorporate any distinguishing characteristics of samples to manage the memory.

Human Workload Reduction. Lastly, we highlight the effectiveness of our method in reducing human workload. In Fig. 8, we plot the human intervention sample ratio for every round, *i.e.*, the percentage of intervention samples in all samples per round. We compare the results for the HITL methods, Ours and IWR. We see that the human intervention ratio decreases over rounds for both methods, as policy performance increases

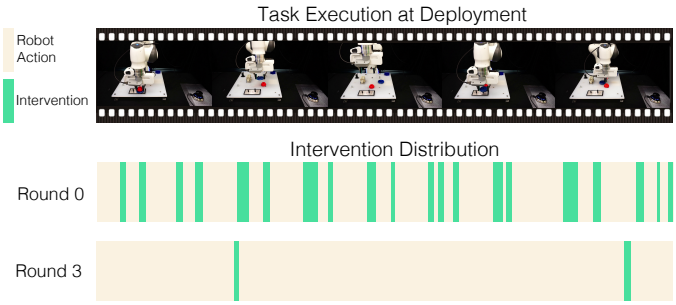

Fig. 9: **Human Intervention Distribution.** The two color bars represent the time duration over 10 consecutive trajectories and whether each step is autonomous robot action (yellow) or human intervention (green). In Round 1, much human intervention is needed to handle difficult situations. In Round 3, the policy needs very little human intervention, and the robot can run autonomously most of the time.

over time. Furthermore, we see that this reduction in human workload is greater for our method compared to IWR.

Qualitatively, we visualize how the division of work of the human-robot team evolves in Figure 9. For the Gear Insertion task, we do 10 trials of task execution in sequence for our method in Round 0 and Round 3, respectively, and record the time duration for human intervention needed during the deployment. Comparing Round 0 and Round 3, the policy in Round 3 needs very little human intervention, and the intervention duration is also much shorter. This confirms the effectiveness of our framework in human workload reduction.

Limitations. Our human-in-the-loop experiment of each task is only conducted with a single human operator. The results can be biased toward the individual's skills, familiarity with the system, and level of risk tolerance. A more extensive human study would enhance our understanding of how human's trust and subjectivity are manifested in time, criteria, and duration of interventions. Furthermore, to ensure trustworthy execution, our current system still requires the human to constantly monitor the robot. Incorporating automated runtime monitoring and error detection strategies [22, 40] would further reduce the human's mental burden. Lastly, for the study of human workload reduction, we employed a simplistic way of measuring human workload based on the percentage of intervention. Conducting in-depth human studies to measure human mental workload would provide deeper insights.

VI. Conclusion

We introduce Sirius, a framework for human-in-the-loop robot manipulation and learning at deployment that both guarantees reliable task execution and also improves autonomous policy performance over time. We utilize the properties and assumptions of human-robot collaboration to develop an intervention-based weighted behavioral cloning method for effectively using deployment data. We also design a practical system that trains and deploys new models continuously under memory constraints. For future work, we would like to improve the flexibility and adaptability of the human-robot shared autonomy, including more intuitive control interfaces and faster policy learning from human feedback. Another direction for future research is alleviating the human cognitive burdens of monitoring and teleoperating the system. Deployment monitoring would be an exciting research direction, allowing the system to automatically detect robot errors without constant human supervision.

Acknowledgment

We thank Ajay Mandlekar for having multiple insightful discussions, and for sharing well-designed simulation task environments and codebases during development of the project. We thank Yifeng Zhu for valuable advice and system infrastructure development for real robot experiments. We would like to thank Tian Gao, Jake Grigsby, Zhenyu Jiang, Ajay Mandlekar, Braham Snyder, and Yifeng Zhu for providing helpful feedback for this manuscript. We acknowledge the support of the National Science Foundation (1955523, 2145283), the Office of Naval Research (N00014-22-1-2204), and Amazon.

References

[1] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "Opal: Offline primitive discovery for accelerating offline reinforcement learning," in *ICLR*, 2021.

[2] M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," vol. 39, 2018, pp. 20–3.

[3] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences," in *IJRR*, vol. 41, SAGE Publications Sage UK: London, England, 2022, pp. 45–67.

[4] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *ICML*, 2019.

[5] S. Cabi *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," in *arXiv preprint arXiv:1909.12200*, 2019.

[6] C. Celemin *et al.*, "Interactive imitation learning in robotics: A survey," 1-2, vol. 10, 2022, pp. 1–197.

[7] E. Chisari, T. Welschehold, J. Boedecker, W. Burgard, and A. Valada, "Correct me if i am wrong: Interactive learning for robotic manipulation," in *RAL*, vol. 7, 2021, pp. 3695–3702.

[8] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, *Deep reinforcement learning from human preferences*, 2017.

[9] C. A. Cruz and T. Igarashi, "A survey on interactive reinforcement learning: Design principles and open challenges," in *DIS*, 2020.

[10] Y. Cui, P. Koppol, H. Admoni, R. G. Simmons, A. Steinfeld, and T. Fitzgerald, "Understanding the relationship between interactions and outcomes in human-in-the-loop machine learning," in *IJCAI*, 2021.

[11] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, "Active reward learning," in *RSS*, 2014.

[12] A. Dragan and S. Srinivasa, "Formalizing assistive teleoperation," in *RSS*, 2012.

[13] A. Dragan and S. Srinivasa, "A policy-blending formalism for shared control," 7, vol. 32, SAGE Publications Sage UK: London, England, 2013, pp. 790–805.

[14] P. Florence *et al.*, "Implicit behavioral cloning," in *CoRL*, 2021.

[15] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," in *arXiv preprint arXiv:1802.01744*, 2020.

[16] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *ICML*, 2019.

[17] K. Gandhi, S. Karamcheti, M. Liao, and D. Sadigh, "Eliciting compatible demonstrations for multi-human imitation learning," in *CoRL*, 2022.

[18] D. Gopinath, S. Jain, and B. D. Argall, "Human-in-the-loop optimization of shared autonomy in assistive robotics," in *RAL*, 2017.

[19] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *NeurIPS*, 2013.

[20] C. Gulcehre *et al.*, "Rl unplugged: Benchmarks for offline reinforcement learning," in *NeurIPS*, 2020.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[22] R. Hoque, A. Balakrishna, E. R. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, "Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning," in *CoRL*, 2021.

[23] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," in *RSS*, vol. 2015, 2015.

[24] D. Kalashnikov *et al.*, "QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation," in *CoRL*, 2018.

[25] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "HG-DAgger: Interactive imitation learning with human experts," in *ICRA*, 2019, pp. 8077–8083.

[26] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," in *NeurIPS*, 2020.

[27] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *K-CAP*, 2009.

[28] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *ICLR*, 2021.

[29] A. Kumar, J. Hong, A. Singh, and S. Levine, "When should we prefer offline reinforcement learning over behavioral cloning?" In *ICLR*, 2022.

[30] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," in *NeurIPS*, 2020.

[31] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," 47, vol. 5, Science Robotics, 2020.

[32] K. Lee, L. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," in *ICML*, 2021.

[33] J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, and S. Legg, "Scalable agent alignment via reward modeling: A research direction," in *arXiv preprint arXiv:1811.07871*, 2018.

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

[35] J. MacGlashan *et al.*, "Interactive learning from policy-dependent human feedback," in *ICML*, 2017, pp. 2285–2294.

[36] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," in *RSS*, 2020.

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

[38] A. Mandlekar *et al.*, "Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data," in *ICRA*, 2020.

[39] A. Mandlekar *et al.*, "What matters in learning from offline human demonstrations for robot manipulation," in *CoRL*, 2021.

[40] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "Ensembledagger: A bayesian approach to safe imitation learning," in *IROS*, 2019, pp. 5041–5048.

[41] K. Muelling *et al.*, "Autonomy infused teleoperation with application to bci manipulation," 2015.

[42] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," in *arXiv preprint arXiv:2006.09359*, 2021.

[43] C. Perez-D'Arpino and J. A. Shah, "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification," in *ICRA*, 2015, pp. 6175–6182.

[44] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NeurIPS*, 1989.

[45] S. Reddy, S. Levine, and A. D. Dragan, "Shared autonomy via deep reinforcement learning," 2018.

[46] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011, pp. 627–635.

[47] F. Sasaki and R. Yamashina, "Behavioral cloning from noisy demonstrations," in *ICLR*, 2021.

[48] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices," vol. 5, 2017, pp. 3909–3943.

[49] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine, "Cog: Connecting new skills to past experience with offline reinforcement learning," in *CoRL*, 2020.

[50] J. Spencer *et al.*, "Learning from interventions: Human-robot interaction as both explicit and implicit feedback," in *RSS*, 2020.

[51] M. Stiber, R. Taylor, and C.-M. Huang, "Modeling human response to robot errors for timely error detection," in *IROS*, 2022, pp. 676–683.

[52] W. Tan *et al.*, "Intervention aware shared autonomy," 2021.

[53] X. Wang, K. Lee, K. Hakhamaneshi, P. Abbeel, and M. Laskin, "Skill preferences: Learning to extract and execute robotic skills from human feedback," in *CoRL*, PMLR, 2022, pp. 1259–1268.

[54] Z. Wang *et al.*, "Critic regularized regression," vol. 33, 2020, pp. 7768–7778.

[55] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone, "Deep tamer: Interactive agent shaping in high-dimensional state spaces," in *AAAI*, vol. 32, 2018.

[56] H. Xu, X. Zhan, H. Yin, and H. Qin, "Discriminator-weighted offline imitation learning from suboptimal demonstrations," in *ICML*, PMLR, 2022, pp. 24 725–24 742.

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

[58] T. Yu *et al.*, "Mopo: Model-based offline oolicy optimization," in *NeurIPS*, 2020.

[59] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone, "Leveraging human guidance for deep reinforcement learning tasks," in *IJCAI*, 2019.

[60] R. Zhang *et al.*, "Human gaze assisted artificial intelligence: A review," in *IJCAI*, Survey track, 2020, pp. 4951–4958.

[61] T. Zhang *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.

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

[63] K. Zolna *et al.*, "Offline learning from demonstrations and unlabeled experience," in *CoRR*, 2020.

VII. Appendix

A. Task Details

We elaborate on the four tasks in this section, providing more details of the task setups, the bottleneck regions, and how they are challenging. The two simulation tasks, Nut Assembly and Tool Hang, are from the robomimic codebase [39] for better benchmarking.

Nut Assembly. The robot picks up a square rod from the table and inserts the rod into a column. The bottleneck lies in grasping the square rod with the correct orientation and turning it such that it aims at the column correctly.

Tool Hang. The robot picks up a hook piece, inserts it into a tiny hole, and then hangs a wrench on the hook. As noted in robomimic [39], this task requires very precise and dexterous control. There are multiple bottleneck regions: picking up the hook piece with the correct orientation, inserting the hook piece with high precision in both position and orientation, picking out the wrench, and carefully aiming the tiny hole at the hook.

Gear Insertion. We design the task scene setup adapting from the common NIST board benchmark[1] Task Board 1, which is designed for standard industrial tasks like peg insertion and electrical connector insertions. Initially, one blue gear and one red gear are placed at a randomized region on the board. The robot picks up two gears in sequence and inserts each onto the gear shafts respectively. The gears' holes are very small, requiring precise insertion on the gear shafts.

Coffee Pod Packing. We design this task for a food manufacturing setting where the robot packs real coffee pods[2] into a coffee pod holder[3]. The robot first opens the coffee pod holder drawer, grasps a coffee pod placed on a random initial position on the table, places the coffee pod into the pod holder, and closes the drawer. The pod holder contains holes that fit precisely to the coffee pods' side, so it requires precise insertion of the coffee pods into the holes. The common bottlenecks are exactly grasping the coffee pod, exact insertion, and releasing the drawer whenever the opening and closing actions are done without getting stuck.

The objects in all tasks are initialized randomly within an x-y position range and with a rotation on the z-axis. The configurations of the simulation tasks follow that in robomimic. We present the reset initialization configuration in Table I for reference.

B. Human-Robot Teaming

We illustrate the actual human-robot teaming process during human-in-the-loop deployment in Figure 10. The robot executes a task (*e.g.*, gear insertion) by default while a human supervises the execution. In this gear insertion scenario, the expected robot behavior is to pick up the gear and insert it down the gear shaft. When the human detects undesirable robot behavior (*e.g.*, gear getting stuck), the human intervenes by taking over control of the robot. The human directly passes in action commands to perform the desired behavior. When the human judges that the robot can continue the task, the human passes control back to the robot.

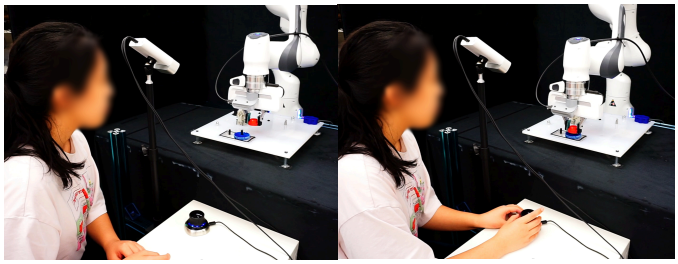

Fig. 10: **Human Robot Teaming.** Left: The robot executes the task by default while a human supervises the execution. Right: When the human detects undesirable robot behavior, the human intervenes.

To enable effective shared human control of the robot, we seek a teleoperation interface that (1) enables humans to control the robot effectively and intuitively and (2) switches between robot and human control immediately once the human decides to intervene or pass the control back to the robot. To this end, we employ SpaceMouse[4] control. The human operator controls a 6-DoF SpaceMouse and passes the position and orientation of the SpaceMouse as action commands. The user can pause when monitoring the computer screen by pressing a button, exert control until the robot is back to an acceptable state, and pass the control back to the robot by stopping the motion on the SpaceMouse.

C. Observation and Action Space

The observational space of all our tasks consists of the workspace camera image, the eye-in-hand camera image, and low-dimensional proprioceptive information. For simulation tasks, we use the operational space controller (OSC) that has a 7D action space; for real-world tasks, we use OSC yaw controller that has a 5D action space.

The minor differences for the Tool Hang task from robomimic [39] default image observation: We use an image size of 128×128 instead of the default 224×224 for training efficiency. Due to the task's need for high-resolution image inputs, we adjust the workspace camera angle to give more details on the objects. This compensates for the need for large image size and boosts policy performance.

Details on low-dimensional proprioceptive information: For simulation tasks, we have the end effector position (3D) and orientation (4D), as well as the distance of the gripper (2D). We have joint positions (7D) and gripper width (1D) for real-world tasks.

The action space of simulation tasks is 7 dimensions in total: x-y-z position (3D), yaw-pitch-roll orientation (3D), and the gripper open-close command $\{1., -1.\}$ (1D). The action space of real-world tasks is 5 dimensions in total: x-y-z position (3D), yaw orientation (1d), and the gripper open-close command $\{1., -1.\}$ (1D).

[1]https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly

[2]https://www.amazon.com/gp/product/B00I5FWWPI

[3]https://www.amazon.com/gp/product/B07D7M93ZW

[4]https://3dconnexion.com/us/spacemouse/

D. Method Implementations

We describe the policy architecture details initally introduced in Section IV-D. Our codebase is based on robomimic [39], a recent open-source project that benchmarks a range of learning algorithms on offline data. We standardize all methods with the same state-of-the-art policy architectures and hyperparameters from robomimic. The architectural design includes ResNet-18 image encoders, random cropping for image augmentation, GMM head, and the same training procedures. The list of hyperparameter choices is presented in Table II. For all BC-related methods, including Ours, IWR, and BC-RNN, we use the same BC-RNN architecture specified in Table III.

For all tasks except for Tool Hang, we use the same hyperparameters with image size 84×84. We use 128×128 for Tool Hang due to its need for high-precision details. We use a few demonstrations for each task to warm-start the policy; the number ranges from 30 to 80 so that the initial policy can all have some level of reasonable behavior regardless of task difficulty. See Table V for all task-dependent hyperparameters.

For IQL [28], we reimplemented the method in our robomimic-based codebase to keep the policy backbone and common architecture the same across all methods. Our implementation is based on the publicly available PyTorch implementation of IQL[5].

We follow the paper's original design with some slight modifications. In particular, the original IQL uses the sparse reward setting where the reward is based on task success. We add a denser reward for IQL to incorporate information on human intervention. To mimic the intervention-guided weights for IQL, we use the following rewards: $r = 1.0$ upon task success, $r = 0.25$ for intervention states, $r = -0.25$ for pre-intervention states, and $r = 0$ for all other states. We found that this version of IQL outperforms the default sparse reward setting. We list the hyperparameters for IQL baseline in Table IV.

E. HITL System Policy Updates

We elaborate on our design choice for HITL system policy update rules discussed in Section V-B of the main paper.

In a practical human-in-the-loop deployment system, there can be many possible design choices for the condition and frequency of policy updates. A few straightforward ones among various designs are: update every specific amount of elapsed time, update after the robot completes a certain number of tasks, or update after human interventions reach a certain number. Our experiments aim to provide a fair comparison between various human-in-the-loop methods and benchmark our method against prior baselines. For consistent evaluation, we follow round updates rules by prior work [25, 37]: 3 rounds of update when the number of intervention samples reaches $1/3$ of the human demonstration samples. The motivation is to evaluate prior baselines in their original setting to ensure fair comparison; moreover, we want to ensure all methods get the same amount of human samples per round. Since they are human-in-the-loop methods, the amount of human samples is important to their utilization. How policies are updated could be a dimension of human-in-the-loop system design on its own right and could be further explored in future work.

[5]https://github.com/rail-berkeley/rlkit/tree/master/examples/iql

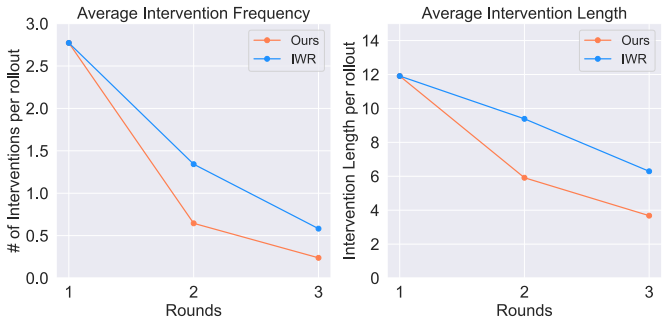

Fig. 11: **More Intervention Behavior Metrics (Nut Assembly).** We present two more metrics to measure human workload over time: average intervention frequency (Left) and average intervention length (Right). We show that our method results in a larger reduction of both metrics over round updates, developing better human trust and human-robot partnership.

F. Human Workload Reduction

We present more results on the effectiveness of our method in reducing human workload as discussed in the main paper. We note that there are different metrics to evaluate human workload, such as the number of control switches and lengths of interventions, as introduced in prior work [22]. We include two additional human workload metrics:

Average intervention frequency: the number of intervention occurrences divided by the number of rollouts. This reflects the number of context switches, *i.e.*, shifts of control between the human and the robot. A higher number of context switches imposes higher concentration and exhaustion on the human.

Average intervention length: length of each intervention in terms of the number of timesteps. This reflects the ease of every intervention - longer intervention occurrence means a higher mental workload to the human for taking control of the robot.

We note that these metrics also reflect the human trust level for the robot. The human makes a decision during robot control: should I intervene at this point? Furthermore, during human control: is the robot in a state where I can safely return control to the robot? Lower intervention frequency and shorter intervention length reflect that human trusts the robot more so that they can intervene at fewer places and return control to the robot faster.

We present the results in Figure 11 using Nut Assembly as an example. We can see that, like the human intervention ratio, the average intervention frequency, and the intervention length decrease. Our method also has a faster reduction of both metrics over round updates. This shows that our human-in-the-loop system fosters good human trust in the robot and develops better human-robot partnerships.

TABLE I: Task objects configuration

Tasks and Objects	Position (x-y)	Orientation (z)
Nut Assembly		
square nut	0.5cm $\times$11.5cm	2π
ToolHang		
hook	2cm $\times$ 2cm	$\pi/9$
wrench	2cm $\times$ 2cm	$\pi/9$
Gear Insertion (Real)		
blue gear	12cm $\times$ 12cm	2π
red gear	12cm $\times$ 12cm	2π
Coffee Pod Packing (Real)		
coffee pod	16cm $\times$ 16cm	2π

TABLE II: Common hyperparameters

Hyperparameter	**Value**
GMM number of modes	5
Image encoder	ResNet-18
Random crop ratio	90% of image height
Optimizer	Adam
Batch size	16
# Training steps per epoch	500
# Total training epochs	1000
Evaluation checkpoint interval (in epoch)	50

TABLE III: BC backbone hyperparameters

Hyperparameter	**Value**
RNN hidden dim	1000
RNN sequence length	10
# of LSTM layers	2
Learning rate	1e$-$4

TABLE IV: IQL hyperparameters

Hyperparameter	**Value**
Reward scale	1.0
Termination	false
Discount factor r	0.99
Beta β	1.0
Adv filter	exponential
V function quantile	0.75
Actor lr	1e$-$4
Actor lr decay factor	0.1
Actor mlp layers	$[1024, 1024]$
Critic lr	1e$-$4
Critic lr decay factor	0.1
Critic mlp layers	$[1024, 1024]$

TABLE V: Task hyperparameters

Hyperparameter	**Nut Assembly**	**ToolHang**	**Gear Insertion (Real)**	**Coffee Pod Packing (Real)**
Image size ($h \times w$)	84×84	128×128	84×84	84×84
Initial # of human demonstrations	50	80	30	30
Evaluation rollout length	400	700	1000	1000