Fleet-DAgger: Interactive Robot Fleet Learning with Scalable Human Supervision

Ryan Hoque, Lawrence Yunliang Chen, Satvik Sharma,
Karthik Dharmarajan, Brijen Thananjeyan, Pieter Abbeel, Ken Goldberg

University of California, Berkeley

Abstract:
Commercial and industrial deployments of robot fleets at Amazon, Nimble, Plus One, Waymo, and Zoox query remote human teleoperators when robots are at risk or unable to make task progress. With continual learning, interventions from the remote pool of humans can also be used to improve the robot fleet control policy over time. A central question is how to effectively allocate limited human attention. Prior work addresses this in the single-robot, single-human setting; we formalize the *Interactive Fleet Learning (IFL)* setting, in which multiple robots interactively query and learn from multiple human supervisors. We propose Return on Human Effort (ROHE) as a new metric and Fleet-DAgger, a family of IFL algorithms. We present an open-source IFL benchmark suite of GPU-accelerated Isaac Gym environments for standardized evaluation and development of IFL algorithms. We compare a novel Fleet-DAgger algorithm to 4 baselines with 100 robots in simulation. We also perform a physical block-pushing experiment with 4 ABB YuMi robot arms and 2 remote humans. Experiments suggest that the allocation of humans to robots significantly affects the performance of the fleet, and that the novel Fleet-DAgger algorithm can achieve up to $8.8\times$ higher ROHE than baselines. See https://tinyurl.com/fleet-dagger for supplemental material.

Keywords: Fleet Learning, Interactive Learning, Human-Robot Interaction

1 Introduction

Amazon, Nimble, Plus One, Waymo, and Zoox use remote human supervision of robot fleets in applications ranging from self-driving taxis to automated warehouse fulfillment [1, 2, 3, 4, 5]. These robots intermittently cede control during task execution to remote human supervisors for corrective interventions. The interventions take place either during learning, when they are used to improve the robot policy, or during execution, when the policy is no longer updated but robots can still request human assistance when needed to improve reliability. In the *continual learning* setting, these occur simultaneously: the robot policy has been deployed but continues to be updated indefinitely with additional intervention data. Furthermore, any individual robot can share its intervention data with the rest of the fleet. As opposed to robot swarms that must coordinate with each other to achieve a common objective, a robot *fleet* is a set of independent robots simultaneously executing the same control policy in parallel environments. We refer to the setting of a robot fleet learning via interactive requests for human supervision (see Figure 1) as *Interactive Fleet Learning (IFL)*.

Of central importance in IFL is the supervisor allocation problem: how should limited human supervision be allocated to robots in a manner that maximizes the throughput of the fleet? Prior work studies this in the single-robot, single-human case. A variety of interactive learning algorithms have been proposed that estimate quantities such as uncertainty [6], novelty [7, 8, 9], risk [8, 9], and predicted action discrepancy [10, 11]. However, it remains unclear which algorithms are the most effective when generalized to the multi-robot, multi-human case.

To this end, we formalize the IFL problem and present the IFL Benchmark (IFLB), a new open-source Python toolkit and benchmark for developing and evaluating human-to-robot allocation algorithms
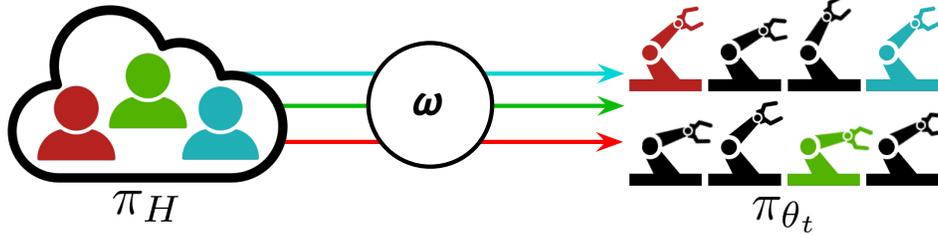

Figure 1: In the *Interactive Fleet Learning (IFL)* setting, a set of M remote human supervisors are allocated to a fleet of N robots ($N \gg M$) with a robot-gated allocation policy ω. The humans share control policy π_H and the robot fleet shares control policy π_{θ_t}, which learns from new human intervention data over time (Section 3).

for fleet learning. The IFLB includes environments from Isaac Gym [12], which enabled efficient simulation of thousands of learning robots for the first time in 2021. This paper makes the following contributions: (1) the first formalism for multi-robot, multi-human interactive learning, (2) the Return on Human Effort (ROHE) metric for evaluating IFL algorithms, (3) the IFLB, an open-source software benchmark and toolkit for IFL algorithms with 3 Isaac Gym environments for complex robotic tasks, (4) Fleet-DAgger, a novel family of IFL algorithms for supervisor allocation, (5) results from large-scale simulation experiments with a fleet of 100 robots, and (6) real robot results with 4 physical robot arms and 2 human supervisors providing teleoperation remotely over the Internet.

2 Related Work

2.1 Allocating Human Supervisors to Robots at Execution Time

For human-robot teams, deciding when to transfer control between robots and humans during execution is a widely studied topic in the literature of both sliding autonomy [13, 14, 15] and Human-Robot Interaction (HRI). In sliding autonomy, also known as adjustable autonomy [16, 17] or adaptive automation [18], humans and robots dynamically adjust their level of autonomy and transfer control to each other during execution [15, 18]. Since identifying which robot to assist in a large robot fleet can be overwhelming for a human operator [19, 20, 21, 22, 23], several strategies, such as using a cost-benefit analysis to decide whether to request operator assistance [13] and using an advising agent to filter robot requests [24], have been proposed to improve the performance of human-robot teams [22, 24, 25] and increase the number of robots that can be controlled [26], a quantity known as "fan-out" [27]. Other examples include user modeling [13, 14, 22, 25] and studying interaction modes [28] for better system and interface design [29, 30]. Zheng et al. [31] propose computing the estimated time until stopping for mobile robots and prioritizing robots accordingly. Ji et al. [32] consider the setting where physical assistance is required to resume tasks for navigation robots and formalize single-human, multi-robot allocation as graph traversal. Dahiya et al. [33] formulate the problem of multi-human, multi-robot allocation during execution as a Restless Multi-Armed Bandit problem. Allocation of humans to robots has also been studied from the perspectives of queueing theory and scheduling theory [23, 34, 35, 36, 37, 38]. The vast majority of the human-robot teaming and queueing theory work, however, does not involve learning; the robot control policies are assumed to be fixed. In contrast, we study supervisor allocation during robot learning, where allocation affects not only human burden and task performance but also the efficiency of policy learning.

2.2 Single-Robot, Single-Human Interactive Learning

Imitation learning (IL) is a paradigm of robot learning in which a robot uses demonstrations from a human to initialize and/or improve its policy [39, 40, 41, 42, 43, 44]. However, learning from purely offline data often suffers from distribution shift [45, 46], as compounding approximation error leads to states that were not visited by the human. This can be mitigated with online data collection with algorithms such as Dataset Aggregation (DAgger) [45] and interactive imitation learning [47, 48, 49]. Human-gated interactive IL algorithms [50, 51, 52] require the human to monitor the robot learning process and decide when to take and cede control of the system. While intuitive, these approaches are not scalable to large fleets of robots or the long periods of time involved in continual learning, as humans cannot effectively focus on many robots simultaneously [21, 22, 23] and are prone to fatigue [53]. To reduce the burden on the supervisor, several robot-gated interactive IL algorithms such as SafeDAgger [10], EnsembleDAgger [6], LazyDAgger [11], and ThriftyDAgger [9] have been proposed, in which the robot actively solicits human interventions when certain criteria are met.

Interactive reinforcement learning (RL) [54, 55, 56, 57, 58] is another active area of research in which robots learn from both online human feedback and their own experience. However, these interactive learning algorithms are designed for and primarily studied in the single-robot, single-human setting. Other works related to single-robot interactive learning include task allocation [59]; in contrast, we focus on efficient robot learning of a single control policy.

2.3 Multi-Robot Interactive Learning

In this paper, we study allocation policies for multiple humans and multiple robots. While many existing works [60, 61, 62, 63, 64] have leveraged NVIDIA's Isaac Gym's [12] capability of parallel simulation to accelerate reinforcement learning with multiple robots, these approaches do not involve human supervision. The work that is closest to ours is by Swamy et al. [65], who study the multi-robot, single-human problem of allocating the attention of one human operator during robot fleet learning. They propose to learn an internal model of human preferences as a human supervises a small fleet of 4 robots and use this model to assist the human in supervising a larger fleet of 12 robots. While this approach mitigates the scaling issue in human-gated interactive IL, even a small fleet of robots can be difficult for a single human supervisor to simultaneously monitor and control.

To the best of our knowledge, this work is the first to formalize and study multi-robot, multi-human interactive learning. This problem setting poses unique challenges, especially as the size of the fleet grows large relative to the number of humans, as each human allocation affects both the robot that receives supervision and the robots that do not receive human attention.

3 Interactive Fleet Learning Problem Formulation

We consider a fleet of N robots operating in parallel as a set of N independent Markov decision processes (MDPs) $\{\mathcal{M}_i\}_{i=1}^N$ specified by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, p_i^0)$ with the same state space $\mathcal{S}$, action space $\mathcal{A}$, unknown transition dynamics $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and discount factor $\gamma \in [0, 1)$, but potentially different initial state distributions p_i^0. We assume the MDPs have an identical indicator function $c(s) : \mathcal{S} \rightarrow \{0, 1\}$ that identifies which states $s \in \mathcal{S}$ violate a *constraint* in the MDP. States that violate MDP constraints are fault states from which the robot cannot make further progress. For instance, the robot may be stuck on the side of the road or have incurred hardware damage. We assume that the timesteps are synchronized across all robots and that they share the same non-stationary policy $\pi_{\theta_t} : \mathcal{S} \rightarrow \mathcal{A}$, parameterized by θ_t at each timestep t.

The collection of $\{\mathcal{M}_i\}_{i=1}^N$ can be reformulated as a single MDP $\mathcal{M} = (\mathcal{S}^N, \mathcal{A}^N, \bar{p}, \bar{r}, \gamma, \bar{p}^0)$, composed of vectorized states and actions of all robots in the fleet (denoted by bold font) and joint transition dynamics. In particular, $\mathbf{s} = (s_1, ..., s_N) \in \mathcal{S}^N, \mathbf{a} = (a_1, ..., a_N) \in \mathcal{A}^N, \bar{p}(\mathbf{s}^{t+1}|\mathbf{s}^t, \mathbf{a}^t) = \Pi_{i=1}^N p(s_i^{t+1}|s_i^t, a_i^t), \bar{r}(\mathbf{s}, \mathbf{a}) = \Sigma_{i=1}^N r(s_i, a_i)$, and $\bar{p}^0 = \Pi_{i=1}^N p_i^0(s_i^0)$.

We assume that robots can query a set of $M \ll N$ human supervisors for assistance interactively (i.e., during execution of π_{θ_t}). We assume that each human can help only one robot at a time and that all humans have the same policy $\pi_H : \mathcal{S} \rightarrow \mathcal{A}_H$, where $\mathcal{A}_H = \mathcal{A} \cup \{R\}$ and R is a *hard reset*, an action that resets the MDP to the initial state distribution $s^0 \sim p_i^0$. As opposed to a *soft reset* that can be performed autonomously by the robot via a reset action $r \in \mathcal{A}$ (e.g., a new bin arrives in an assembly line), a *hard reset* requires human intervention due to constraint violation (i.e., entering some s where $c(s) = 1$). A human assigned to a robot either performs hard reset R (if $c(s) = 1$) or teleoperates the robot system with policy π_H (if $c(s) = 0$). A hard reset R takes t_R timesteps to perform, and all other actions take 1 timestep.

Supervisor allocation (i.e., the assignment of humans to robots) is determined by an allocation policy

$$\omega : (\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t) \mapsto \boldsymbol{\alpha}^t \in \{0, 1\}^{N \times M} \quad \text{s.t.} \quad \sum_{j=1}^M \boldsymbol{\alpha}_{ij}^t \leq 1 \text{ and } \sum_{i=1}^N \boldsymbol{\alpha}_{ij}^t \leq 1 \quad \forall i, j, \quad (1)$$

where $\mathbf{s}^t$ are the current states for each of the robots, $\boldsymbol{\alpha}^t$ is an $N \times M$ binary matrix that indicates which robots will receive assistance from which human at the current timestep t, and $\mathbf{x}^t$ is an augmented state containing any auxiliary information for each robot, such as the type and duration of an ongoing intervention. Unlike Dahiya et al. [33] which studies execution-time allocation, the allocation policy ω here depends on the current robot policy π_{θ_t}, which in turn affects the speed of the policy learning. While there are a variety of potential objectives to consider, e.g., minimizing constraint violations in a safety-critical environment, we define the IFL objective as *return on human*

effort (ROHE):

$$\max_{\omega \in \Omega} \mathbb{E}_{\tau \sim p_{\omega, \theta_0}(\tau)} \left[\frac{M}{N} \cdot \frac{\sum_{t=0}^{T} \bar{r}(\mathbf{s}^t, \mathbf{a}^t)}{1 + \sum_{t=0}^{T} \|\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)\|_F^2} \right], \qquad (2)$$

where Ω is the set of allocation policies, T is the total amount of time the fleet operates (rather than the time horizon of an individual task execution), θ_0 are the initial parameters of the robot policy, and $\| \cdot \|_F$ is the Frobenius norm. The objective is the expected ratio of the cumulative reward across all timesteps and all robots to the total amount of human time spent helping robots with allocation policy ω, with a scaling factor to normalize for the number of robots and humans and an addition of 1 in the denominator for the degenerate case of zero human time. Intuitively, the ROHE measures the performance of the robot fleet normalized by the total human effort required to achieve this performance. We provide a more thorough derivation of the ROHE objective in Appendix 8.1.

Since human teleoperation with π_H provides additional online data, this data can be used to update the robot policy π_{θ_t} with some policy update function f (e.g., gradient descent):

$$\begin{cases} D^{t+1} \leftarrow D^t \cup D_H^t \text{ where } D_H^t := \{(s_i^t, \pi_H(s_i^t)) : \pi_H(s_i^t) \neq R \text{ and } \sum_{j=1}^{M} \boldsymbol{\alpha}_{ij}^t = 1\} \\ \pi_{\theta_{t+1}} \leftarrow f(\pi_{\theta_t}, D^{t+1}) \end{cases} \qquad (3)$$

4 Interactive Fleet Learning Algorithms

4.1 Fleet-DAgger

Given the problem formulation above, we propose Fleet-DAgger, a family of IFL algorithms, where an *IFL algorithm* is a supervisor allocation strategy (i.e., it specifies an $\omega \in \Omega$ as defined in Section 3). The learning algorithm in Fleet-DAgger is interactive imitation learning with dataset aggregation from prior work [45, 50, 9]: its policy update function f is supervised learning on D^t, which consists of all human data collected so far (Section 3). The novel component of each Fleet-DAgger algorithm is its supervisor allocation scheme based on unique *priority function* $\hat{p} : (s, \pi_{\theta_t}) \rightarrow [0, \infty)$ that indicates a priority score to assign to each robot based on its state s and the current policy π_{θ_t}, where, similar to scheduling theory, a higher value indicates a higher priority robot. To reduce thrashing [11, 9], Fleet-DAgger algorithms also specify t_T, the minimum time a human supervisor must spend teleoperating a robot.

Fleet-DAgger uses priority function $\hat{p}$ and t_T to define an allocation ω as follows (see Appendix 8.2 for the full pseudocode). At each timestep t, Fleet-DAgger first scores all robots with $\hat{p}$ and sorts the robots by their priority values. If a human supervisor is currently performing hard reset action R and t_R timesteps have not elapsed, that human continues to help that robot. If a human is currently teleoperating a robot and the minimum t_T timesteps have not elapsed, that human continues to teleoperate the robot. If a robot with a human supervisor continues to be high priority after the minimum intervention time (t_R for a hard reset or t_T for teleoperation) has elapsed, that human remains assigned to the robot. If a human is available to help a robot, the human is reassigned to the robot with the highest priority value that is currently unassisted. Finally, if a robot has priority $\hat{p}(\cdot) = 0$, it does not receive assistance even if a human is available.

4.2 Fleet-DAgger Algorithms

All algorithms below specify a unique priority function $\hat{p}$, which is synthesized with Fleet-DAgger as described in Section 4.1 to specify an allocation ω. More details are available in the appendix.

Constraint (C): The Constraint baseline measures the performance of the robot fleet when only trained on offline human demonstrations (i.e., $\forall t$, $\pi_{\theta_t} = \pi_{\theta_0}$). At all timesteps t, this baseline gives priority $\hat{p}(\cdot) = 1$ for robots that have violated a constraint ($c(s_i^t) = 1$) and require a hard reset, and $\hat{p}(\cdot) = 0$ for all other robots. We refer to this as C-prioritization for Constraint. Thus, the robot fleet can only receive hard resets from human supervisors (no human teleoperation). Without C-prioritization, robots that require hard resets would remain indefinitely idle.

Random: This baseline simply assigns a random priority for each robot at each timestep. To control the total amount of human supervision, we introduce a threshold hyperparameter such that if a robot's priority value is below the threshold, its priority is set to zero and it will not request help.

Fleet-EnsembleDAgger (U.C.): This baseline adapts EnsembleDAgger [6] to the IFL setting. En-sembleDAgger uses the output variance among an ensemble of neural networks bootstrapped on

subsets of the training data as an estimate of epistemic uncertainty; accordingly, we define the robot priority for Fleet-EnsembleDAgger as ensemble variance. Since ensemble variance is designed for continuous action spaces, for environments with discrete action spaces we instead estimate the uncertainty with the Shannon entropy [66] among the outputs of a single classifier network.We refer to this priority function as U-prioritization for Uncertainty. Finally, since EnsembleDAgger was not designed for environments with constraint violations and idle robots will negatively affect the ROHE, we add C-prioritization for a more fair comparison. Specifically, given an uncertainty threshold value, robots with uncertainty above threshold are prioritized first in order of their uncertainty (U), followed by constraint-violating robots (C).

Fleet-ThriftyDAgger (U.G.C.): This baseline adapts the ThriftyDAgger algorithm [9] to the IFL setting. ThriftyDAgger uses a synthesis of uncertainty (which we refer to as the U-prioritization value) and the probability of task failure (G-prioritization, estimated with a Goal critic Q-function) to query a human for supervision. Since Fleet-DAgger requires a single metric by which to compare different robots, we adapt ThriftyDAgger to the fleet setting by calculating a linear combination of the U value and G value after normalizing each value with running estimates of their means and standard deviations. As in [9], we pretrain the goal critic on an offline dataset of human and robot task execution. Similar to Fleet-EnsembleDAgger, we first prioritize by the combined uncertainty-goal value above a parameterized threshold, followed by C-prioritization.

Constraint-Uncertainty-Risk (C.U.R.): Here we propose a novel Fleet-DAgger algorithm. As the name suggests, C.U.R. does C-prioritization, followed by U-prioritization, followed by R-prioritization. R stands for Risk, which we define as the probability of constraint violation. Intuitively, idle robots should be reset in order to continue making progress, uncertain robots should receive more human supervision in areas with little to no reference behavior to imitate, and robots at risk should request human teleoperation to safety before an expensive hard reset. As in [67], we estimate the probability of constraint violation with a safety critic Q-function and initialize the safety critic on an offline dataset of constraint violations. C.U.R. also prioritizes differently at the beginning of execution for a parameterized length of time, during which constraint violations are assigned *zero priority* rather than high priority. Here, the intuition is that rather than attending to hard resets for an initially low-performing policy, human intervention should instead be spent on valuable teleoperation data that can improve the robot policy. Hence, during the initial period, constraint-violating robots remain idle and human attention is allocated to the teleoperation of a smaller number of robots.

5 Interactive Fleet Learning Benchmark

While many algorithms have been proposed for interactive learning [11, 6, 9, 10], to our knowledge there exists no unified benchmark for evaluating them. To facilitate reproducibility and standardized evaluation for IFL algorithms, we introduce the Interactive Fleet Learning Benchmark (IFLB). The IFLB is an open-source Python implementation of IFL with a suite of simulation environments and a modular software architecture for rapid prototyping and evaluation of new IFL algorithms.

5.1 Environments

The IFLB is built on top of NVIDIA Isaac Gym [12], a highly optimized software platform for end-to-end GPU-accelerated robot learning released in 2021, without which the simulation of hundreds of learning robots would be computationally intractable. The IFLB can run efficiently on a single GPU and currently supports the following 3 Isaac Gym environments with high-dimensional continuous state and action spaces (see Figure 2): (1) **Humanoid**, a bipedal legged locomotion task from OpenAI Gym [68], (2) **Anymal**, a quadruped legged locomotion task with the ANYmal robot by ANYbotics, and (3) **AllegroHand**, a task involving dexterous manipulation of a cube with a 4-finger Allegro Hand by Wonik Robotics. Constraint violation is defined as (1) the humanoid falling down, (2) the ANYmal falling down on its torso or knees, and (3) dropping the cube from the hand, respectively. End users can also add their own custom Isaac Gym environments with ease.

5.2 Software Architecture

The IFLB defines 3 interfaces for the development of IFL algorithms: (1) agents, (2) supervisors, and (3) allocations. An *agent* is an implementation of the robot fleet policy π_{θ_t} (Section 3), such as an IL or RL agent. A *supervisor* is an implementation of the supervisor policy π_H (Section 3), such as a fully trained RL agent, a model-based planner, or a teleoperation interface for remote human supervisors. Lastly, an *allocation* is an implementation of the priority function $\hat{p}$ (Section 4), such as
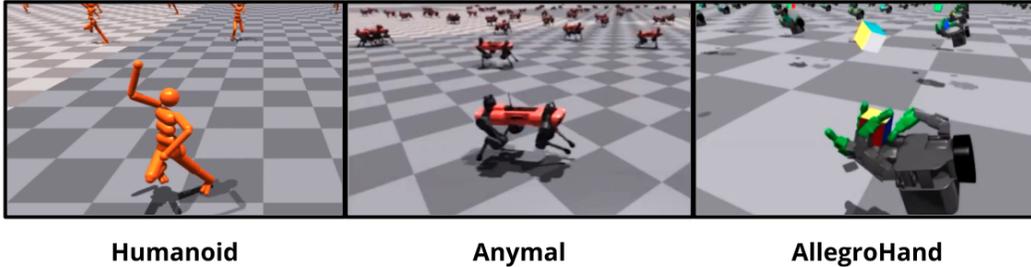
Figure 2: Isaac Gym benchmark environments in the IFLB.

C.U.R. priority or ThriftyDAgger priority. For reference, the IFLB includes an imitation learning agent, a fully trained RL supervisor using Isaac Gym's reference PPO [69] implementation, and all allocations from Section 4, which we use in our experiments. Users of the IFLB can flexibly implement their own IFL algorithms by defining new agents, supervisors, and allocations.

Given an agent, supervisor, allocation, and environment, the IFLB runs Fleet-DAgger as described in Section 4.1. IFLB allows flexible command line configuration of all parameters of the experiment (e.g., t_T, t_R, N, M) as well as the parameters of the agent, supervisor, and allocation. If desired, the code can also be modified to support families of IFL algorithms other than Fleet-DAgger. The benchmark is available open-source at https://github.com/BerkeleyAutomation/ifl_benchmark.

6 Experiments

6.1 Metrics

Throughout online training, we measure four metrics at each timestep t: (1) the cumulative number of successful task completions across the fleet and up to time t; (2) cumulative hard resets (i.e., constraint violations); (3) cumulative idle time, i.e., how long robots spend idle in constraint-violating states waiting for hard resets; and (4) the return on human effort (ROHE, Equation 2), where reward is a sparse $r \in \{0, 1\}$ for successful task completion and cumulative human time is measured in hundreds of timesteps. For the Humanoid and Anymal locomotion environments, success is defined as reaching the episode horizon without constraint violation and with reward of at least 95% of that of the supervisor policy. For the goal-conditioned tasks, i.e., AllegroHand and the physical block-pushing task, success is defined by reaching the goal state.

6.2 IFLB Simulation Experiments

Experimental Setup: We evaluate all Fleet-DAgger algorithms in the 3 benchmark simulation environments: Humanoid, Anymal, and AllegroHand. We use reinforcement learning agents fully trained with PPO [69] as the algorithmic supervisor π_H. We initialize the robot policy π_{θ_0} with behavior cloning on an offline dataset of 5000 state-action pairs. For a fair comparison, the Constraint baseline is given additional offline data equal to the average amount of human time solicited by C.U.R. by operation time boundary T. The Random baseline's priority threshold is set such that in expectation, it reaches the average amount of human time solicited by C.U.R. by time T. Since Fleet-ThriftyDAgger requires a goal-conditioned task, it is only evaluated on AllegroHand. All training runs are executed with $N = 100$ robots, $M = 10$ humans, $t_T = 5$, $t_R = 5$, and operation time $T = 10000$, and are averaged over 3 random seeds. In the appendix, we provide additional experiments including ablation studies on each component of the C.U.R. algorithm and an analysis of hyperparameter sensitivity to the number of humans M, minimum intervention time t_T, and hard reset time t_R. The IFLB code provides instructions for reproducing results.

Results: We plot results in Figure 3. First, we observe that the choice of IFL algorithm has a significant impact on all metrics in all environments, indicating that allocation matters in the IFL setting. We also observe that the robot fleet achieves a higher throughput (number of cumulative task successes) with C.U.R. allocation than baselines in all environments at all times. C.U.R. also attains a higher ROHE, indicating more efficient use of human supervision. An increase in ROHE over time signifies that the improvement in the robot policy π_{θ_t} outpaces cumulative human supervision, indicating that the IFL algorithms learn not only where to allocate humans but also when to *stop* requesting unnecessary supervision. C.U.R. also incurs fewer hard resets than baselines, especially
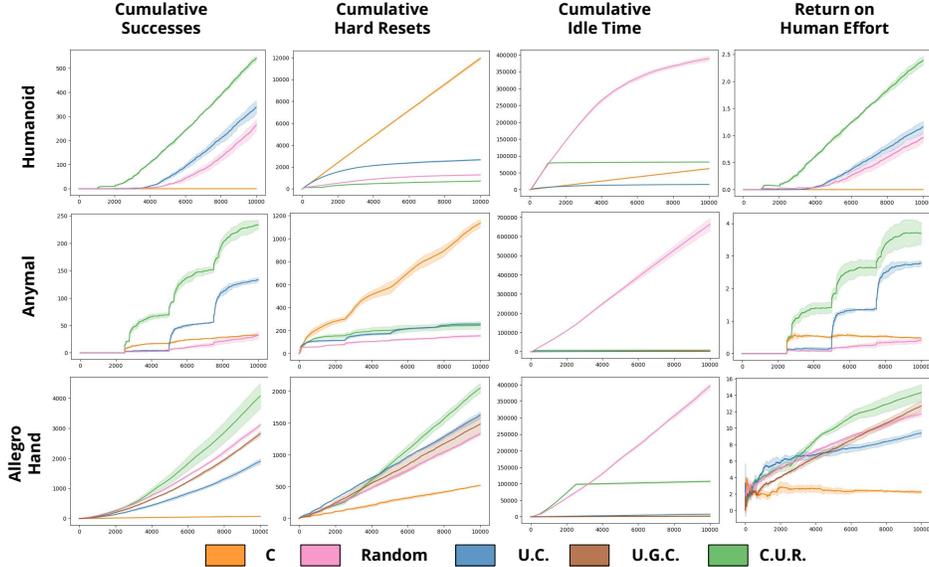
Figure 3: Simulation results in the IFLB with $N = 100$ robots and $M = 10$ human supervisors, where the x-axis is timesteps from 0 to $T = 10,000$. Shading indicates 1 standard deviation. The C.U.R. algorithm outperforms all baselines on all environments in terms of ROHE and cumulative successes. (Note that the shape of the Anymal curves is due to its success classification, episode horizon of 2500, and low hard resets.)
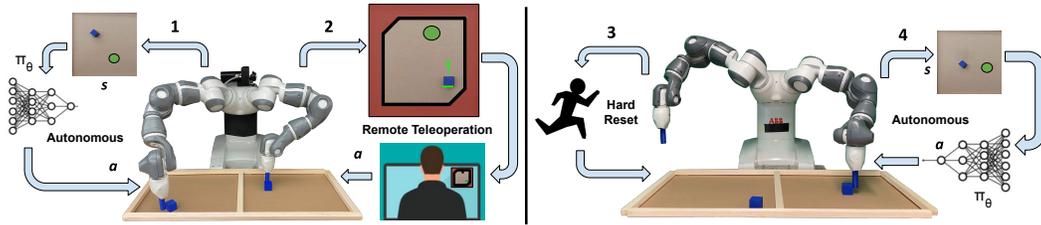

Figure 4: **Physical Task Setup:** an example timestep t in the physical experiment with 2 humans and 4 independent identical robot arms each executing the block pushing task. **Robot 1** queries robot policy π_{θ_t} for an action given an overhead image of the workspace and executes it in the workspace. **Robot 2** is teleoperated by a remote Human 1, where the human views the overhead image and specifies a pushing action through a user interface. The red region at the edges of the workspace are constraint violation regions. Human 2 is performing a physical hard reset for **Robot 3**, which has violated a constraint in a previous timestep. **Robot 4** autonomously executes the same robot policy as that of Robot 1 on its own state.

Constraint, which must constantly hard reset robots with a low-performing offline policy. For AllegroHand, however, C.U.R. incurs higher hard resets and a smaller ROHE margin over baselines. We hypothesize that since the task is too challenging to execute without human supervision in the given fleet operation time, prioritizing hard resets ironically only gives the robots additional opportunities to violate constraints. We also see that C-prioritization effectively eliminates idle time; C.U.R. idle time flattens out after the initial period without C-prioritization.

6.3 Physical Block-Pushing Experiment

Experimental Setup: Finally, we evaluate Fleet-DAgger in a physical block-pushing experiment with $N = 4$ ABB YuMi robot arms and $M = 2$ human supervisors. Each robot arm has an identical setup for the block-pushing task that consists of a square wooden workspace, a small blue cube, and a cylindrical end-effector. See Figure 4 for the hardware setup. Constraint violation occurs when the cube hits the boundary or has moved into regions out of reach for the end-effector at two opposite corners of the workspace. The objective of each robot is to reach a goal position randomly sampled from the allowable region of the workspace. At each timestep, the robot chooses one of four discrete pushing actions corresponding to pushing each of the four vertical faces of the cube orthogonally by a fixed distance. The robot policy takes an overhead image observation of the cube in the workspace
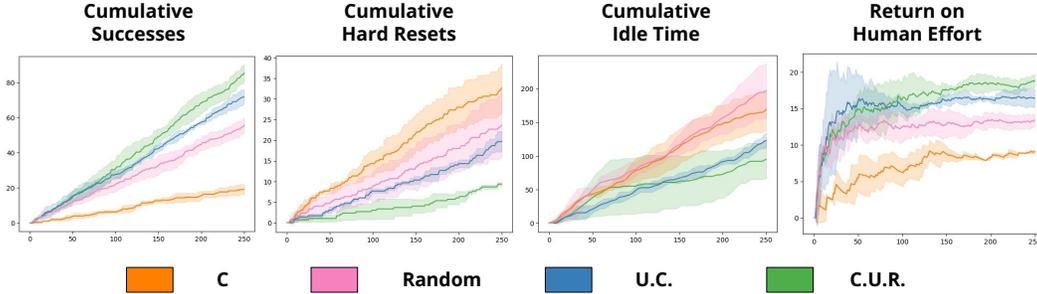
7

Figure 5: Physical results for the block-pushing task with 4 robots and 2 humans, where the x-axis is timesteps. C.U.R. achieves higher ROHE and cumulative successes as well as lower cumulative hard resets and idle time. Shading indicates 1 standard deviation.

with the goal programatically generated in the image. Hard resets are physical adjustments of the cube, while teleoperation is performed over the Internet by a remote human supervisor, who specifies one of the 4 pushing actions via a keyboard interface. We set $t_T = 3$, $t_R = 5$, and $T = 250$ for a total of 4×250 pushing actions per trial and run each algorithm with 3 random seeds. All algorithms are initialized with an offline dataset of 3750 image-action pairs (375 samples with $10\times$ data augmentation).

Results: We plot results in Figure 5. We observe that the C.U.R. algorithm achieves higher ROHE, higher cumulative successes, lower hard resets, and lower idle time than baselines, albeit by a small margin. Results suggest that (1) training an accurate safety critic is more difficult in high-dimensional image space, leading to a smaller gap between C.U.R. and U.C. (i.e., Fleet-EnsembleDAgger), and (2) U-prioritization in its current form is less suitable for real-world multimodal human supervisors than it is for deterministic algorithmic supervisors, resulting in a smaller increase in ROHE over time. Since a human may arbitrarily choose one of multiple equally suitable actions, high robot uncertainty over these actions does not necessarily translate to a need for human supervision.

7   Limitations and Future Work

The IFL formulation has a number of modeling assumptions that limit its generality. (1) The human supervisors are homogeneous, (2) all robots operate in the same state and action space, (3) all robots are independent and do not coordinate with each other, (4) humans have perfect situational awareness [23] and can move to different robots without any switching latency, (5) hard reset time is constant, and (6) timesteps are synchronous without network latency or other communication issues [70]. In terms of experiments, the simulations have algorithmic rather than human supervision, and the physical task is relatively straightforward with discrete planar actions.

In future work, we will work on removing the assumptions above: learning from nonhomogeneous supervisors [71] and collecting data from large fleets with limited network bandwidth [72], for instance. We will also study reinforcement learning algorithms for IFL and run more large-scale physical experiments. We hope that other robotics researchers will develop their own IFL algorithms and evaluate them using the benchmark toolkit to accelerate progress.

References

[1] A. Brown. Robin deals with a world where things are changing all around it. *Amazon Science Blog*, Apr. 2022. URL https://www.amazon.science/latest-news/robin-deals-with-a-world-where-things-are-changing-all-around-it.

[2] J. Smith. Robotic Arms Are Using Machine Learning to Reach Deeper Into Distribution. *Wall Street Journal*, Jan. 2022. ISSN 0099-9660. URL https://tinyurl.com/bdhvfnjc.

[3] Logistics Automation with Plus One Robotics. *Parcel Monitor*, May 2022. URL https://tinyurl.com/4susnhmz.

[4] A. C. Madrigal. Waymo's Robot Cars, and the Humans Who Tend to Them. *The Atlantic*, Aug. 2018. URL https://www.theatlantic.com/technology/archive/2018/08/waymos-robot-cars-and-the-humans-who-tend-to-them/568051/. Section: Technology.

[5] E. Chu. How Zoox Builds Autonomous Vehicles from the Wheels Up-Blog. *AI Exchange*, Apr. 2022. URL https://exchange.scale.com/public/blogs/how-zoox-builds-autonomous-vehicles-from-the-wheels-up.

[6] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer. EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[7] B. Kim and J. Pineau. Maximum mean discrepancy imitation learning. In *Robotics: Science and Systems (RSS)*, 2013.

[8] M. Laskey, S. Staszak, W. Hsieh, J. Mahler, F. Pokorny, A. Dragan, and K. Goldberg. SHIV: Reducing supervisor burden using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016.

[9] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *Conference on Robot Learning (CoRL)*, 2021.

[10] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017.

[11] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg. LazyDAgger: Reducing context switching in interactive imitation learning. In *IEEE International Conference on Automation Sciences and Engineering (CASE)*, 2021.

[12] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning. *ArXiv*, abs/2108.10470, 2021.

[13] B. Sellner, R. Simmons, and S. Singh. User modelling for principled sliding autonomy in human-robot teams. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 197–208. Springer, 2005.

[14] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh. Coordinated multiagent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE*, 94(7):1425–1444, 2006.

[15] M. B. Dias, B. Kannan, B. Browning, E. Jones, B. Argall, M. F. Dias, M. Zinck, M. Veloso, and A. Stentz. Sliding autonomy for peer-to-peer human-robot teams. In *Proceedings of the international conference on intelligent autonomous systems*, pages 332–341, 2008.

[16] P. Scerri, D. V. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *Journal of Artificial Intelligence Research*, 17:171–228, 2002.

[17] D. Kortenkamp, D. Keirn-Schreckenghost, and R. P. Bonasso. Adjustable control autonomy for manned space flight. In *2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484)*, volume 7, pages 629–640. IEEE, 2000.

[18] T. B. Sheridan. Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: Distinctions and modes of adaptation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(4):662–667, 2011.

[19] J. Y. Chen and M. J. Barnes. Supervisory control of multiple robots: Effects of imperfect automation and individual differences. *Human Factors*, 54(2):157–174, 2012.

[20] M. Lewis. Human interaction with multiple remote robots. *Reviews of Human Factors and Ergonomics*, 9(1):131–174, 2013.

[21] S.-Y. Chien, M. Lewis, S. Mehrotra, and K. Sycara. Imperfect automation in scheduling operator attention on control of multi-robots. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 1169–1173. SAGE Publications Sage CA: Los Angeles, CA, 2013.

[22] J. R. Peters, V. Srivastava, G. S. Taylor, A. Surana, M. P. Eckstein, and F. Bullo. Human supervisory control of robotic teams: Integrating cognitive modeling with engineering design. *IEEE Control Systems Magazine*, 35(6):57–80, 2015.

[23] S.-Y. Chien, Y.-L. Lin, P.-J. Lee, S. Han, M. Lewis, and K. Sycara. Attention allocation for human multi-robot control: Cognitive analysis based on behavior data and hidden states. *International Journal of Human-Computer Studies*, 117:30–44, 2018.

[24] A. Rosenfeld, N. Agmon, O. Maksimov, and S. Kraus. Intelligent agent supporting human–multi-robot team collaboration. *Artificial Intelligence*, 252:211–231, 2017.

[25] J. W. Crandall, M. L. Cummings, M. Della Penna, and P. M. De Jong. Computing the effects of operator attention allocation in human control of multiple robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 41(3):385–397, 2010.

[26] S. A. Zanlongo, P. Dirksmeier, P. Long, T. Padir, and L. Bobadilla. Scheduling and path-planning for operator oversight of multiple robots. *Robotics*, 10(2):57, 2021.

[27] D. R. Olsen Jr and S. B. Wood. Fan-out: Measuring human control of multiple robots. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 231–238, 2004.

[28] M. Cakmak, C. Chao, and A. L. Thomaz. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118, 2010.

[29] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

[30] J. Y. Chen and M. J. Barnes. Human–agent teaming for multirobot control: A review of human factors issues. *IEEE Transactions on Human-Machine Systems*, 44(1):13–29, 2014.

[31] K. Zheng, D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita. Supervisory control of multiple social robots for navigation. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 17–24. IEEE, 2013.

[32] T. Ji, R. Dong, and K. Driggs-Campbell. Traversing supervisor problem: An approximately optimal approach to multi-robot assistance. *Robotics: Science and Systems (RSS)*, 2022.

[33] A. Dahiya, N. Akbarzadeh, A. Mahajan, and S. L. Smith. Scalable operator allocation for multi-robot assistance: A restless bandit approach. *IEEE Transactions on Control of Network Systems*, 2022.

[34] R. B. Cooper. Queueing theory. In *Proceedings of the ACM'81 conference*, pages 119–122, 1981.

[35] L. Sha, T. Abdelzaher, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, A. K. Mok, et al. Real time scheduling theory: A historical perspective. *Real-time systems*, 28 (2):101–155, 2004.

[36] M. C. Gombolay, R. J. Wilcox, and J. A. Shah. Fast scheduling of robot teams performing tasks with temporospatial constraints. *IEEE Transactions on Robotics*, 34(1):220–239, 2018.

[37] W. B. Rouse. Adaptive allocation of decision making responsibility between supervisor and computer. In *Monitoring behavior and supervisory control*, pages 295–306. Springer, 1976.

[38] A. Daw, R. C. Hampshire, and J. Pender. How to staff when customers arrive in batches. *arXiv*, abs/1907.12650, 2019.

[39] J. A. Bagnell. An invitation to imitation. Technical Report CMU-RI-TR-15-08, Carnegie Mellon University, Pittsburgh, PA, 3 2015.

[40] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 7 2018.

[41] F. Codevilla, E. Santana, L. A. M., and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *International Conference on Computer Vision*, 2019.

[42] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1), 1991.

[43] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

[44] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[45] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[46] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. DART: Noise injection for robust imitation learning. In *Conf. on Robot Learning (CoRL)*, 2017.

[47] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009.

[48] S. Jauhri, C. Celemin, and J. Kober. Interactive imitation learning in state-space. In *Conference on Robot Learning (CoRL)*. PMLR, 2020.

[49] M. Rigter, B. Lacerda, and N. Hawes. A framework for learning from demonstration with minimal human effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, 2020.

[50] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. HG-DAgger: Interactive imitation learning with human experts. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[51] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Learning from interventions: Human-robot interaction as both explicit and implicit feedback. In *Proc. Robotics: Science and Systems (RSS)*, 2020.

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

[53] R. R. Murphy and E. Rogers. Cooperative assistance for remote robot supervision. *Presence: Teleoperators & Virtual Environments*, 5(2):224–240, 1996.

[54] L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni. Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018.

[55] A. Kurenkov, A. Mandlekar, R. Martin-Martin, S. Savarese, and A. Garg. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. In *Conf. on Robot Learning (CoRL)*, 2019.

[56] F. Wang, B. Zhou, K. Chen, T. Fan, X. Zhang, J. Li, H. Tian, and J. Pan. Intervention aided reinforcement learning for safe and practical policy optimization in navigation. In *Conf. on Robot Learning (CoRL)*, 2018.

[57] G. Kahn, P. Abbeel, and S. Levine. Land: Learning to navigate from disengagements. *IEEE Robotics and Automation Letters*, 6(2):1872–1879, 2021.

[58] S. Trick, F. Herbert, C. A. Rothkopf, and D. Koert. Interactive reinforcement learning with bayesian fusion of multimodal advice. *IEEE Robotics and Automation Letters*, 7(3):7558–7565, 2022.

[59] S. Vats, O. Kroemer, and M. Likhachev. Synergistic scheduling of learning and allocation of tasks in human-robot teams. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2022.

[60] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *Robotics: Science and Systems (RSS)*, 2022.

[61] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[62] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel. Adversarial motion priors make good substitutes for complex reward functions. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[63] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4): 1–17, 2022.

[64] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[65] G. Swamy, S. Reddy, S. Levine, and A. D. Dragan. Scaled autonomy: Enabling human operators to control robot fleets. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5942–5948. IEEE, 2020.

[66] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal (1948)*, 1948.

[67] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. P. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6:4915–4922, 2021.

[68] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.

[69] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[70] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[71] K. Gandhi, S. Karamcheti, M. Liao, and D. Sadigh. Eliciting compatible demonstrations for multi-human imitation learning. *Conference on Robot Learning (CoRL)*, 2022.

[72] O. Akcin, P. Li, S. Agarwal, and S. Chinchali. Data games: A game-theoretic approach to swarm robotic data collection. *Conference on Robot Learning (CoRL)*, 2022.

8 Appendix

8.1 Mathematical Details of the IFL Problem Formulation

Recall that ROHE takes the expectation over a distribution of trajectories, $p_{\omega,\theta_0}(\tau)$, where each trajectory $\tau = (\mathbf{s}^0, \mathbf{a}^0, ..., \mathbf{s}^T, \mathbf{a}^T)$ is composed of consecutive task episodes separated by resets and where the state-action tuples come from both π_θ and π_H. This distribution of trajectories is induced by ω and θ_0 because θ_0 parameterizes the initial robot policy π_{θ_0}, and ω affects the states that comprise D_H^t, which updates the robot policy π_θ for subsequent timesteps. In this section, we derive the mathematical relationship between the trajectory distribution $\tau \sim p_{\omega,\theta_0}(\tau)$ and the allocation policy ω.

Given an allocation policy ω, the human policy π_H, and the robot policy π_{θ_t} at each timestep t, the joint hybrid human-robot policy of all robots can be expressed as

$$
\pi_{H \cup R}^t(\mathbf{s}) = \begin{bmatrix} \pi_{\theta_t}(s_1)(1 - \mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_1}) + \pi_H(s_1)\mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_1} \\ \vdots \\ \pi_{\theta_t}(s_N)(1 - \mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_N}) + \pi_H(s_N)\mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_N} \end{bmatrix}, \quad (4)
$$

where $\mathbb{1}_{(.)}$ is an indicator function that selects the robot policy π_{θ_t} if robot i is allocated to a human and selects the human policy $\pi_H(s)$ otherwise. For notational convenience, $\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_i := \sum_{j=1}^M \boldsymbol{\alpha}_{ij}^t \in \{0, 1\}$.

The trajectory distribution $p_{\omega,\theta_0}(\tau)$ can then be expressed as

$$
p_{\omega,\theta_0}(\tau) = p_{\omega,\theta_0}(\mathbf{s}^0, \mathbf{a}^0, ..., \mathbf{s}^T, \mathbf{a}^T) \quad (5)
$$

$$
= \bar{p}^0(\mathbf{s}^0) \prod_{t=0}^T \pi_{H \cup R}^t(\mathbf{a}^t | \mathbf{s}^t) \prod_{t=0}^{T-1} \bar{p}(\mathbf{s}^{t+1} | \mathbf{s}^t, \mathbf{a}^t). \quad (6)
$$

We comment that the soft and hard reset can be easily incorporated into the transition dynamics $\bar{p}$ depending on the task. For example, for constraint violations (i.e., hard resets) $c(s_i^t) = 1$, we can set $p(s_i^{t+k} | s_i^t, a_i^t) = \delta(s_i^t)$ for $1 \le k \le t_R - 1$ and $p(s_i^{t+T_R} | s_i^t, a_i^t) = p_i^0(s^0)$ where $\delta(\cdot)$ is the Dirac delta function and t_R is the hard reset time. Similarly, for goal-conditioned tasks with goal g, soft resets after achieving the goal can be expressed through the transition dynamics $p(s_i^{t+1} | s_i^t, a_i^t) = p_i^0(s^0)$ if $s_i^t \subseteq g$. For MDPs with finite time horizon where the environment resets when the maximum time horizon is reached, we can augment the state with additional time information that keeps track of the timestep in each episode, and reset the state when it times out. In this case, the MDP transition dynamics will be time-dependent: $p_t(s_i^{t+1} | s_i^t, a_i^t)$.

8.2 Fleet-DAgger Algorithm Details

In this section, we provide a detailed algorithmic description of Fleet-DAgger.

Fleet-DAgger uses priority function $\hat{p}$ and t_T to define an allocation policy ω. Concretely, it can be interpreted as a function F where $F(\hat{p}) = \omega$, i.e., a "meta-algorithm" (algorithm that outputs another algorithm) akin to function composition. The pseudocode of Fleet-DAgger is provided in Algorithm 1.

8.3 Additional Experiment Details

8.3.1 Simulation Hyperparameters

Implementations of C.U.R. and baselines are available in the code supplement, where the scripts are configured to run with the same hyperparameters as the simulation experiments in the main text. Recall that $N = 100$ robots, $M = 10$ humans, minimum intervention time $t_T = 5$, hard reset time $t_R = 5$, and operation time $T = 10000$. For reference, additional parameters are in Table 1, where $|S|$ is the dimensionality of the (continuous) state space, $|A|$ is the dimensionality of the (continuous) action space, $\hat{r}$ is the risk threshold below which robots are assigned zero risk, $\hat{u}$ is the uncertainty threshold below which robots are assigned zero uncertainty, and t_I is the length of the initial C.U.R. period during which constraint violation is not prioritized.

Algorithm 1 Fleet-DAgger

Input: MDP $\mathcal{M}$, Number of robots N, Number of humans M, Priority function $\hat{p}$, Minimum teleoperation time t_T, Hard reset time t_R
Output: Allocation policy ω

1: **function** $\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)$ `#The allocation policy` ω `returns a matrix` $\boldsymbol{\alpha}^t \in \{0,1\}^{N \times M}$
2:      Compute priority scores of each robot: $\hat{p}(s_i^t, \pi_{\theta_t})$ $\forall i = 1, ..., N$
3:      Initialize $\boldsymbol{\alpha}_{ij}^t = 0$ $\forall i, j$
4:      **for** $i \in \{1, \ldots, N\}$ **do**
5:          **for** $j \in \{1, \ldots, M\}$ **do**
6:              **if** $\boldsymbol{\alpha}_{ij}^{t-1} = 1$ **then**   `#For robots that were receiving assistance during the last timestep,`
    `check whether the minimum intervention time has lapsed using auxiliary information` $\mathbf{x}^t$
7:                  **if** Intervention type for robot $i = $ Hard reset and Intervention duration $< t_R$ **then**
8:                      $\boldsymbol{\alpha}_{ij}^t = 1$
9:                  **if** Intervention type for robot $i = $ Teleop and Intervention duration $< t_T$ **then**
10:                    $\boldsymbol{\alpha}_{ij}^t = 1$
11:      Let $I = \{i : \sum_{j=1}^{M} \boldsymbol{\alpha}_{ij}^t = 1\}$ `#Set of robots that will continue with past assistance`
12:      Let $J = \{j : \sum_{i=1}^{N} \boldsymbol{\alpha}_{ij}^t = 1\}$ `#Set of humans that will continue with past assistance`
13:      Sort robot indices with positive priority scores that are not in I from highest to lowest, denoted as $\{i_1, i_2, ...\}$
14:      Let $k = 1$
15:      **for** $j \in \{1, \ldots, M\} \setminus J$ **do**
16:          $\boldsymbol{\alpha}_{i_k,j}^t = 1$
17:          $k = k + 1$
18:      **return** $\boldsymbol{\alpha}$

19: **return** ω

Environment	$|S|$	$|A|$	$\hat{r}$	$\hat{u}$	t_I
Humanoid	108	21	0.5	0.05	1000
Anymal	48	12	0.5	0.05	250
AllegroHand	88	16	0.5	0.15	2500

Table 1: Simulation environment hyperparameters.

8.3.2   Training Critic Q-Functions

Some IFL algorithms require pretraining a safety critic (C.U.R.) or goal critic (Fleet-ThriftyDAgger) to assist in supervisor allocation. Here we provide details on how we train these critics in practice. Additional details about training these critics in practice are also available in Recovery RL [67] and ThriftyDAgger [9] for the safety critic and goal critic respectively.

To collect a dataset of constraint violations, we simply run Behavior Cloning for a fixed amount of timesteps. Intuitively, since the initial robot policy π_{θ_0} is not highly performant, the robot should expect to encounter constraint violations, and these violations will occur within the state distribution visited by the robot fleet in the initial stages of online training. One could also inject noise into the BC policy to induce more constraint violations, or explicitly solicit human demonstrations of constraint violations as noted in [67]. For Humanoid, Anymal, and AllegroHand, we collect a dataset of 19625 transitions with 376 constraint violations, 19938 transitions with 63 constraint violations, and 19954 transitions with 47 constraint violations, respectively. The safety critic is then trained via Q-learning for 3000 gradient steps where a constraint-violating transition can be interpreted as incurring sparse reward $r = 1$ and all other transitions have reward $r = 0$. To reduce class imbalance issues, transitions are sampled from the replay buffer such that constraint-violating samples constitute 25% of the minibatch, which was found to be useful in practice in [67].

To collect a dataset of successes to pretrain the goal critic, we instead run the expert policy π_H, which is more likely to reach the goal. For AllegroHand, we collect a dataset of 19994 transitions with 489 successes. We then pretrain the goal critic in the same manner as the safety critic. Both the safety and goal critic continue to update during online training with the additional constraint violation and success data encountered.

Environment	Algorithm	Successes ($\uparrow$)	Hard Resets ($\downarrow$)	Idle Time ($\downarrow$)
Humanoid	BC	0.0 ± 0.0	11925.3 ± 118.8	62473.7 ± 869.1
	Random	746.3 ± 40.5	340.0 ± 59.2	1700.0 ± 296.1
	Fleet-ED	617.7 ± 66.3	570.3 ± 139.0	2851.7 ± 694.8
	C.U.R.	$\mathbf{771.0 \pm 25.5}$	$\mathbf{289.3 \pm 21.1}$	$\mathbf{1446.7 \pm 105.3}$
	Expert	894	115	575
Anymal	BC	32.7 ± 0.5	1134.3 ± 33.9	5669.7 ± 170.0
	Random	207.3 ± 27.2	232.3 ± 89.6	1162.3 ± 449.1
	Fleet-ED	$\mathbf{257.3 \pm 1.2}$	109.0 ± 16.9	545.0 ± 84.4
	C.U.R.	$\mathbf{257.0 \pm 8.8}$	$\mathbf{64.3 \pm 8.6}$	$\mathbf{321.7 \pm 42.9}$
	Expert	293	1	5
AllegroHand	BC	70.0 ± 5.0	$\mathbf{523.0 \pm 9.4}$	$\mathbf{2614.3 \pm 46.4}$
	Random	$\mathbf{7360.3 \pm 231.0}$	2954.3 ± 131.0	14767.7 ± 653.2
	Fleet-ED	3032.0 ± 191.2	1764.7 ± 225.4	8818.0 ± 1129.7
	Fleet-TD	4296.3 ± 161.7	1397.7 ± 112.6	6988.0 ± 562.9
	C.U.R.	6032.7 ± 236.2	2343.7 ± 82.5	11715.0 ± 411.9
	Expert	21609	1202	6013

Table 2: Robot policy performance for each of the algorithms in Section 6.2. We report cumulative successes, hard resets, and idle time. We do not report return on human effort as teleoperation is not performed for this experiment.

8.3.3   Physical Experiment Protocol

We execute 3 trials of each of 4 algorithms (Behavior Cloning, Random, Fleet-EnsembleDAgger, C.U.R.) on the fleet of 4 robot arms for 250 timesteps each, for a total of $3 \times 4 \times 4 \times 250 = 12000$ individual pushing actions. Human teleoperation and hard resets were performed by the authors, where teleoperation is collected through an OpenCV (https://opencv.org/) graphical user interface and hard resets are physical adjustments of the cube toward the center of the workspace. Each of the 2 ABB YuMi robots is connected via Ethernet to a Linux machine on its local area network; the driver program connects to each machine over the Internet via the Secure Shell Protocol (SSH) to send robot actions and receive camera observations. The 2 YuMis are in different physical locations 0.5 miles apart, and all 4 arms execute actions concurrently with multiprocessing. $10\times$ data augmentation is performed on the initial offline dataset of 375 state-action pairs as well as the online data collected during execution as follows:

- Linear contrast uniformly sampled between 85% and 115%
- Add values uniformly sampled between -10 and 10 to each pixel value per channel
- Gamma contrast uniformly sampled between 90% and 110%
- Gaussian blur with σ uniformly sampled between 0.0 and 0.3
- Saturation uniformly sampled between 95% and 105%
- Additive Gaussian noise with σ uniformly sampled between 0 and $\frac{1}{80} \times 255$

8.3.4   Evaluating Trained Simulation Policies

Here we execute all policies from Section 6.2 after the 10,000 timesteps of online training for an additional 10,000 timesteps *without additional human teleoperation* to evaluate the quality of the learned robot policies in isolation. As in Section 6.2, $N = 100$ robots, $M = 10$ humans (for hard resets only), $t_R = 5$, and $T = 10000$, where allocation is performed by C-prioritization. We also include the expert policy performance (all $N = 100$ robots teleoperated by the trained PPO supervisor for $T = 10000$ steps) in the table for reference.

Results are in Table 2. We find that the policy learned via C.U.R. outperforms baselines and approaches expert-level performance for Humanoid and Anymal, but is second most performant and significantly below expert-level performance for AllegroHand, indicating $T = 10,000$ timesteps is insufficient for learning a highly performant robot policy in this challenging environment.

8.4   Hyperparameter Sensitivity and Ablation Studies

In this section, we run additional simulation experiments in the IFL benchmark to study (1) ablations of the components of the C.U.R. algorithm (Figure 6), (2) sensitivity to the ratio of number of robots

N to number of humans M (Figure 7), (3) sensitivity to minimum intervention time t_T (Figure 8), and (4) sensitivity to hard reset time t_R (Figure 9). All runs are averaged over 3 random seeds, where shading indicates 1 standard deviation.

Ablations: We test C.U.R.(-i), the C.U.R. algorithm without the initial period during which constraint violation is not prioritized. We also test all subsets of the C.U.R. priority function without the initial period. For example, U. indicates only prioritizing by uncertainty, and C.R. indicates prioritizing by constraint violations followed by risk (no uncertainty). Results suggest that C.U.R. outperforms all ablations in all environments in terms of ROHE and cumulative successes and is competitive in terms of hard resets and idle time. However, as in the main text, C.U.R. and C. incur more hard resets in AllegroHand than alternatives, as again, prioritizing constraint violations for a hard environment where learning has not converged may ironically enable more opportunities for hard resets. Interestingly, while C.U.R. outperforms ablations in ROHE in AllegroHand for large T, U-prioritization's ROHE is significantly higher for small values of T. We observe that since U. achieves very low cumulative successes in the same time period, U. must be requesting an extremely small amount of human time early in operation, resulting in erratic ratio calculations.

Number of Humans: While keeping N fixed to 100 robots, we run C.U.R. with default hyperparameters and vary M to be 1, 5, 10, 25, and 50 humans. In the Humanoid and Anymal environment, as expected, cumulative successes increases with the number of humans. The performance boost gets smaller as M increases: runs with 25 and 50 humans have very similar performance. Despite lower cumulative successes, $M = 10$ achieves the highest ROHE, suggesting a larger set of humans provides superfluous interventions. We also observe that with only 1 human, the number of hard resets and idle time is very large, as the human is constantly occupied with resetting constraint-violating robots, which fail at a faster rate than the human can reset them. Finally, in the AllegroHand environment, the number of humans when $M \geq 5$ does not make much of a visible difference, perhaps due to the relatively high number of cumulative successes.

Minimum Intervention Time: We run C.U.R. with default hyperparameters but vary t_T to be 1, 5, 20, 50, 100, and 500 timesteps. We observe that both decreasing t_T from 5 to 1 and increasing t_T to 20 and beyond have a negative impact on the ROHE due to ceding control prematurely (in the former case) and superfluous intervention length (in the latter). Hard resets are low and idle time is high for large t_T as the humans are occupied providing long teleoperation interventions. This also negatively affects throughput, as cumulative successes falls for very large t_T. Long interventions may also be less useful training data, as in the limit these interventions reduce to more offline data (i.e., labels for states encountered under the human policy rather than that of the robot).

Hard Reset Time: Finally, we run C.U.R. with default hyperparameters but vary t_R to be 1, 5, 20, 50, 100, and 500 timesteps. As expected, the ROHE decreases as t_R increases, as more human effort is required to achieve the same return. The other metrics follow similar intuitive trends: increasing t_R results in a decrease in cumulative successes, decrease in hard resets, and increase in idle time.

Other Parameters: We also found that the batch size (256 in our experiments) and number of gradient steps per experiment timestep (1 in our experiments) significantly impact the policy learning speed as well as computation time, and the effect varies with the size of the fleet (N) and set of human supervisors (M). This is because N and M (and ω) determine how much new data is available at each time t, the batch size and number of gradient steps determine how much data to update the policy with at each time t, and updating the policy with backpropagation is the computational bottleneck in the IFLB simulation.
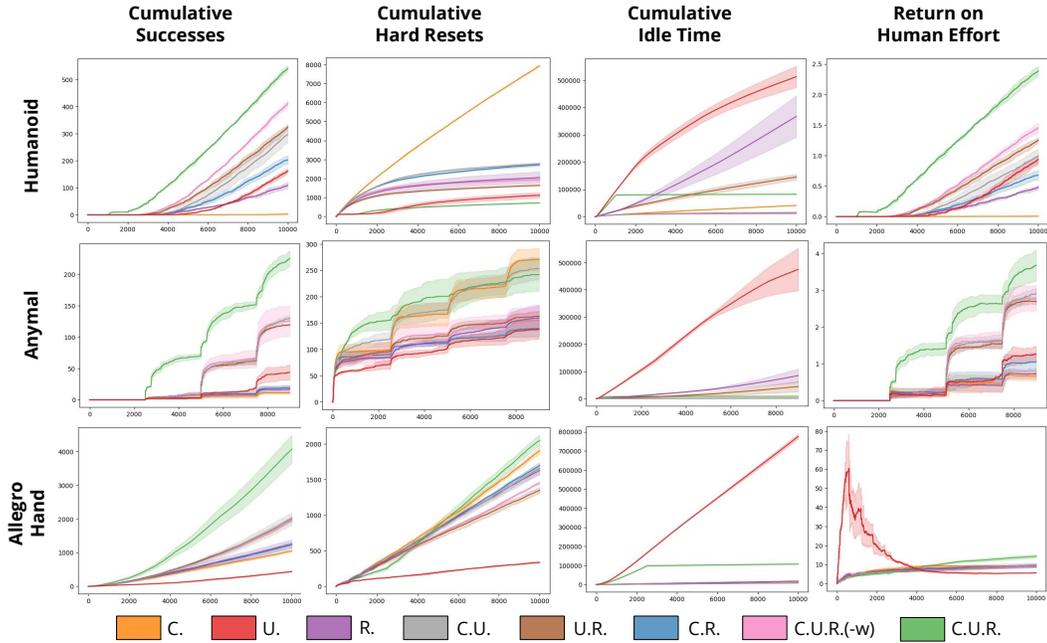
Figure 6: **Ablations:** Simulation results in the Isaac Gym benchmark tasks with ablations of C.U.R., where the x-axis is timesteps from 0 to $T = 10,000$. We plot the metrics described in 6.1. The C.U.R. algorithm outperforms all ablations on all environments in terms of ROHE and cumulative successes (except AllegroHand ROHE for low T values) and is competitive with ablations for cumulative hard resets and idle time.
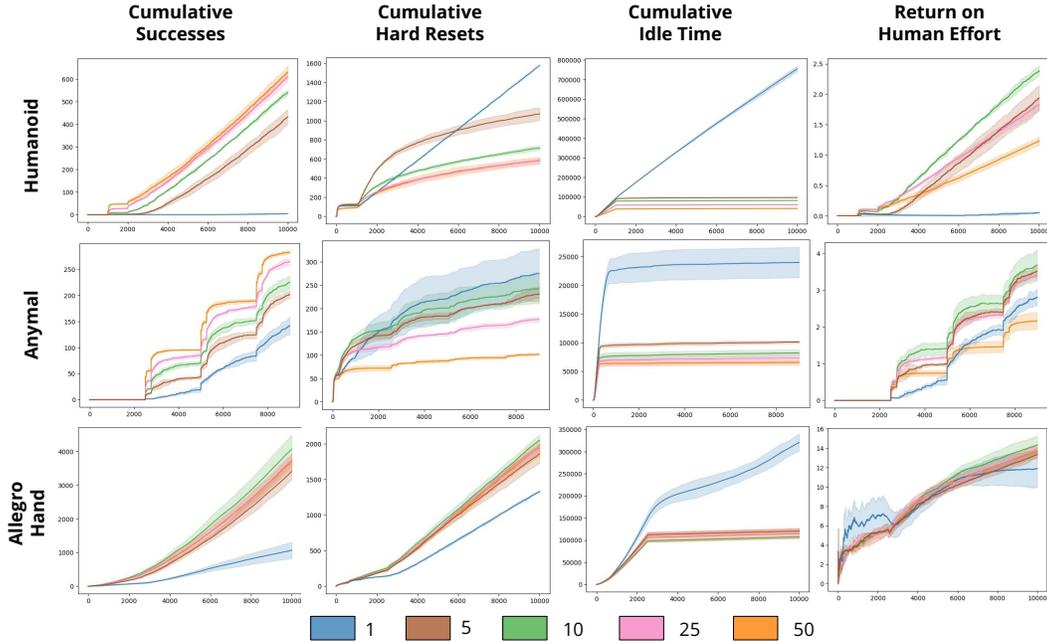

Figure 7: **Number of Humans:** Simulation results in the Isaac Gym benchmark tasks with $N = 100$ robots and M human supervisors, where M varies and the x-axis is timesteps from 0 to $T = 10,000$.
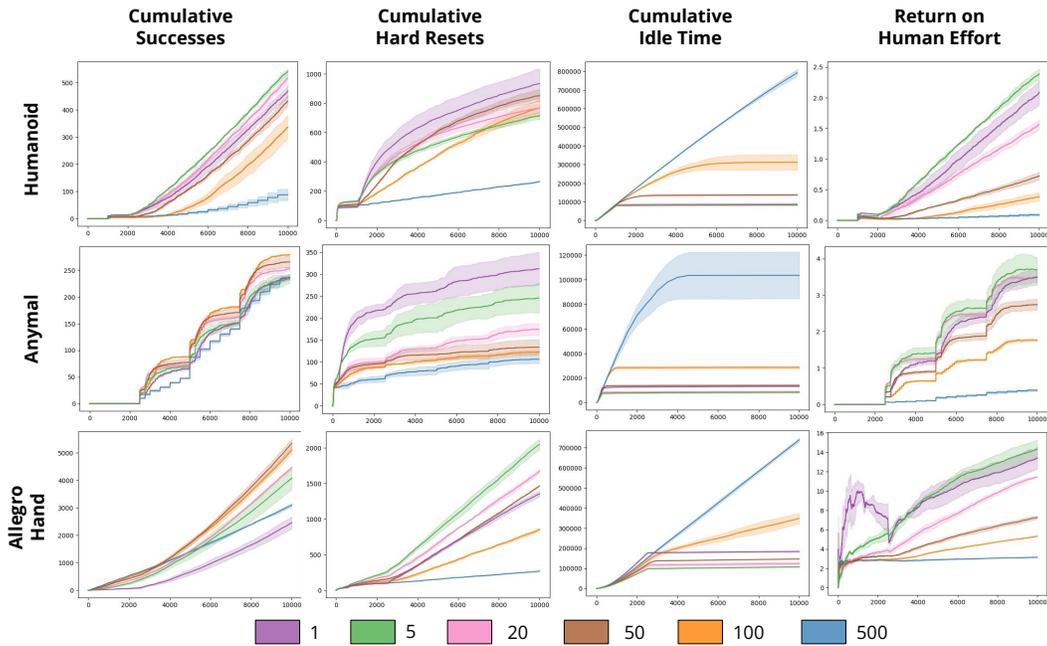
Figure 8: **Minimum Intervention Time:** Simulation results in the Isaac Gym benchmark tasks for variations in minimum intervention time t_T, where the x-axis is timesteps from 0 to $T = 10,000$.
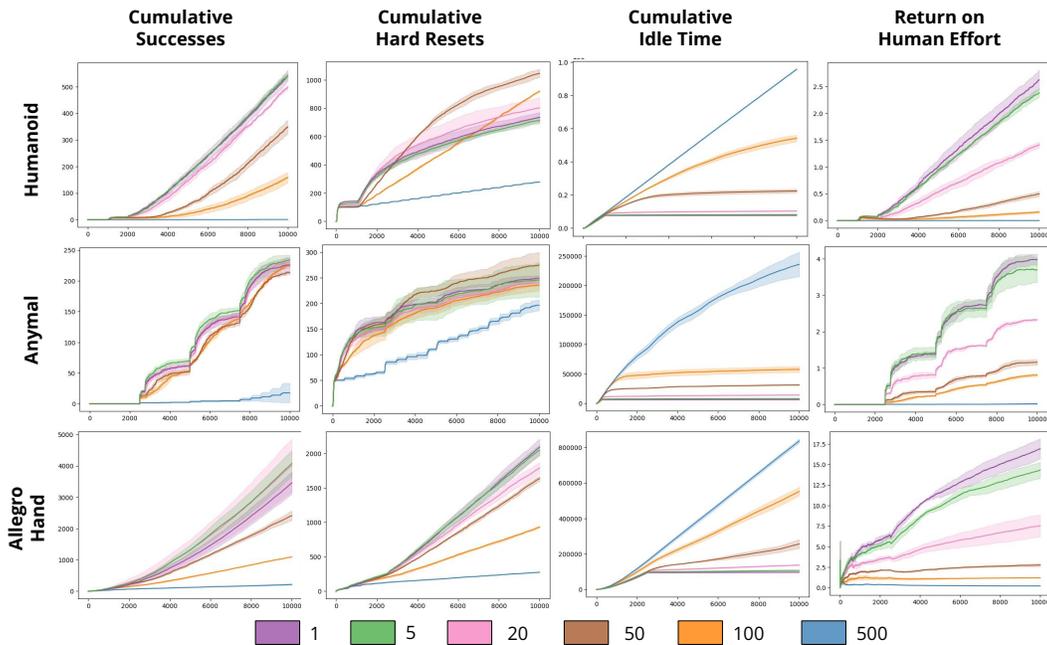

Figure 9: **Hard Reset Time:** Simulation results in the Isaac Gym benchmark tasks for variations in hard reset time t_R, where the x-axis is timesteps from 0 to $T = 10,000$.