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

Anonymous Author(s)
Affiliation
Address
email

Abstract:

Commercial and industrial deployments of robot fleets often fall back on remote human teleoperators during execution 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 to individual robots. 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 present a fully implemented open-source IFL benchmark suite of GPU-accelerated Isaac Gym environments for the evaluation of IFL algorithms. We propose Fleet-DAgger, a family of IFL algorithms, and compare a novel Fleet-DAgger algorithm to 4 baselines in simulation. We also perform 1000 trials of a physical block-pushing experiment with 4 ABB YuMi robot arms. Experiments suggest that the allocation of humans to robots significantly affects the performance of the fleet, and that our algorithm achieves up to 8.8× higher return on human effort than baselines. See https://sites.google.com/view/fleet-dagger for supplemental material.

Keywords: Fleet Learning, Interactive Learning, Human Robot Interaction

1 Introduction

Waymo, Zoox, Amazon, Nimble Robotics, and Plus One Robotics use remote human supervision of robot fleets in applications ranging from self-driving taxis to automated warehouse fulfillment [1, 2, 3, 4]. 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place either during learning, when they are used to improve the robot policy, or at execution time, when the policy is no longer updated but robots can still request human assistance when needed to improve reliability.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 for the same task in parallel environments. We refer to the setting of a robot fleet learning via interactive requests for human supervision 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 [5], novelty [6, 7, 8], risk [7, 8], and predicted action discrepancy [9, 10]. However, it remains unclear which algorithms are the most effective when generalized to the multi-human, multi-robot case.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

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. The IFLB includes environments from Isaac Gym [11], which enabled efficient simulation of thousands of learning robots for the first time in 2021. This paper makes the following contributions: (1) a formalism for interactive fleet learning, (2) the IFLB, an open-source software benchmark and toolkit for IFL algorithms with 3 Isaac Gym environments for complex robotic tasks, (3) Fleet-DAgger, a novel family of IFL algorithms, (4) results from large-scale simulation experiments with a fleet of 100 robots, and (5) results from 1000 physical trials 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 [12, 13, 14] and Human-Robot Interaction (HRI). In sliding autonomy, also known as adjustable autonomy [15, 16] or adaptive automation [17], humans and robots dynamically adjust their level of autonomy and transfer control to each other during execution [14, 17]. Since identifying which robot to assist in a large robot fleet can be overwhelming for a human operator [18, 19, 20, 21, 22], several strategies, such as using a cost-benefit analysis to decide whether to request operator assistance [12] and using an advising agent to filter robot requests [23], have been proposed to improve the performance of human-robot teams [23, 21, 24] and increase the number of robots that can be controlled [25], a quantity known as "fan-out" [26]. Other examples include user modeling [12, 13, 21, 24] and studying interaction modes [27] for better system and interface design [28, 29]. Zheng et al. [30] propose to compute the time until stopping for each robot based on its estimated risk and prioritize the robots accordingly. Ji et al. [31] 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. [32] formulate the problem of multi-human, multi-robot allocation during robot execution as a Restless Multi-Armed Bandit problem. In addition, allocating humans to address robot requests has also been studied from the perspective of queueing theory and scheduling theory [33, 34, 22, 35, 36, 37]. 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 the effectiveness of allocation policies affects not only human burden and the performance of human-robot team but also the efficiency of policy learning.

2.2 Single-Human, Single-Robot 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 [38, 39, 40, 41, 42, 43]. However, learning from purely offline data often suffers from distribution shift [44, 45], 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 Aggregration (DAgger) [44] and interactive imitation learning [46, 47, 48]. Human-gated interactive IL algorithms [49, 50, 51] 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 [20, 21, 22] and are prone to fatigue [52]. To reduce the burden on the supervisor, several robot-gated interactive IL algorithms such as SafeDAgger [9], EnsembleDAgger [5], LazyDAgger [10], and ThriftyDAgger [8] have been proposed, in which the robot actively solicits human interventions when certain criteria are met. Interactive reinforcement learning (RL) [53, 54, 55, 56] 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 studied in the single-human, single-robot setting. Other works related to single-robot interactive learning include task allocation [57]; in contrast, we focus on efficient robot learning of a single task.

2.3 Multi-Robot Interactive Learning

In this paper, we study allocation policies for multiple humans and multiple robots. While many existing works [58, 59, 60, 61, 62] have leveraged NVIDIA's Isaac Gym's [11] capability of parallel simulation to accelerate reinforcement learning with multiple robots, these approaches consider robots in isolation without interactive human supervision. The work that is closest to ours is by Swamy et al. [63], 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, a small fleet of robots can be difficult for a single human supervisor to optimally control. In contrast, we consider robot-gated algorithms for requesting and allocating supervision from multiple humans.

To the best of our knowledge, this work is the first to formalize and study multi-human, multi-robot 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 and identical Markov decision processes (MDPs) $\{\mathcal{M}_i\}_{i=1}^N$ specified by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ with the same state space $\mathcal{S}$, action space $\mathcal{A}$, unknown transition dynamics $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor $\gamma \in [0, 1)$, and initial state distribution p_0. We assume the MDPs have an identical indicator function $c(s) : \mathcal{S} \to \{0, 1\}$ 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 $\pi_{\theta_t} : \mathcal{S} \to \mathcal{A}$, parameterized by θ_t at each timestep t, where θ_t is updated over time with continual learning.

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

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 $\pi_H : \mathcal{S} \to \mathcal{A}_H$, 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 $s^0 \sim p_0$. 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

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

where $\mathbf{s}^t$ are the current states for each of the robots, $\boldsymbol{\alpha}^t$ is an $M \times N$ 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. [32] 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)*:

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

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. 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}:

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

where f is a policy update function (e.g., gradient descent).

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 an algorithm for supervisor allocation for IFL (i.e., it specifies an $\omega \in \Omega$ as defined in Section 3). Fleet-DAgger seeks to maximize the IFL objective in the context of interactive imitation learning with dataset aggregation: its policy update function f is supervised learning on D^t, which consists of all human data collected so far (Section 3). In addition, each Fleet-DAgger algorithm defines a unique *priority function* $\hat{p} : (s, \pi_{\theta_t}) \rightarrow [0, \infty)$ that indicates how much 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 [10, 8], 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. 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. We include the pseudocode of Fleet-DAgger in the supplemental material.

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.

Behavior Cloning: At all timesteps t, this baseline gives priority $\hat{p}(\cdot) = 1$ for robots that have violated a constraint (i.e., $c(s_i^t) = 1$) 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 (i.e., no human teleoperation). Without C-prioritiziation, robots that require hard resets would remain indefinitely idle and not learn.

Random: This baseline simply assigns a random priority for each robot at each timestep. Additionally, 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 This baseline adapts EnsembleDAgger [5] to the IFL setting. EnsembleDAgger 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 [64] 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, followed by constraint-violating robots.

Fleet-ThriftyDAgger: This baseline adapts the ThriftyDAgger algorithm [8] 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 (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-prioritization value and the probability of task failure after normalizing each value with running estimates of their means and standard deviations. As in [8], 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 (prioritize all constraint-violating robots), followed by U-prioritization (prioritize uncertain robots above a minimum threshold), followed by R-prioritization, where 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 [65], we estimate the probability of constraint violation with a safety critic Q-function. We initialize the safety critic on an offline dataset of constraint violations.In addition, we implement an initial warmup period 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 warmup 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 [10, 5, 8, 9], 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 [11], 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 currently supports the following 3 Isaac Gym environments with high-dimensional continuous state and action spaces: (1) **Humanoid**, a bipedal legged locomotion task from OpenAI Gym [66], (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. See Figure 1 for images of each of these tasks. While these three tasks are the current fully supported environments in the IFLB, adding new Isaac Gym

Figure 1: Isaac Gym benchmark environments in the IFLB.

environments to the IFLB is straightforward and will be done by the maintainers on a rolling basis (and can also easily be done by end users).

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 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 [67] 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. A code sample for reproducing simulation results is available in the supplement and a public codebase will be released on Github after the anonymous review period.

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), approximated as the ratio of cumulative task successes to the cumulative amount of human time (both hard resets and teleoperation) in hundreds of timesteps for simulation experiments and tens of timesteps for physical experiments. We do not measure ROHE for Behavior Cloning as it is an offline algorithm. 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 AllegroHand, a goal-conditioned task, success is defined by reaching a goal state. For the physical block-pushing experiment (Section 6.3), a robot achieves 1 success for a push in the correct direction and 5 successes for reaching the goal.

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 [67] 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 Behavior Cloning 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

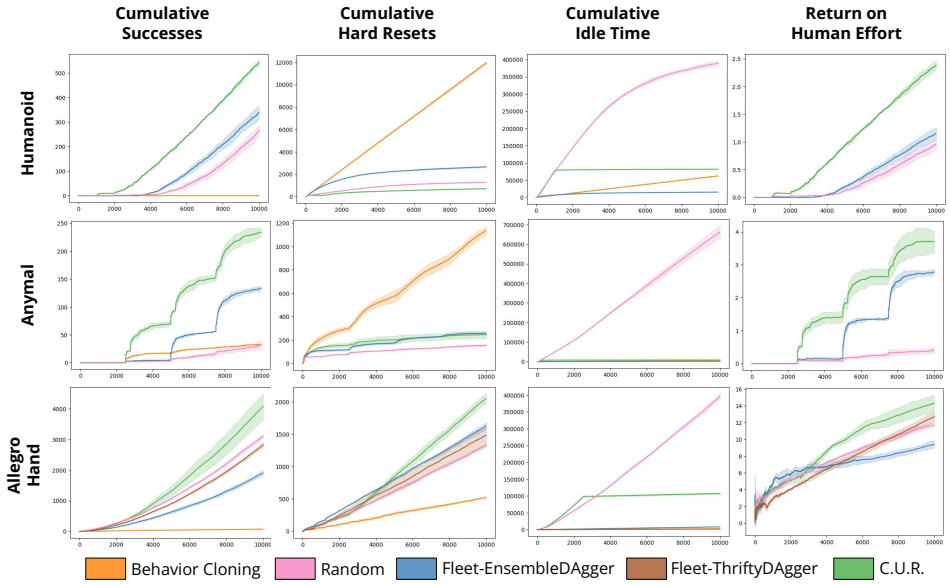

Figure 2: Simulation results in the Isaac Gym benchmark tasks with $N = 100$ robots and $M = 10$ human supervisors, where the x-axis is timesteps from 0 to $T = 10,000$. We plot the metrics described in 6.1. The C.U.R. algorithm outperforms all baselines on all environments in terms of ROHE and cumulative successes and is competitive with baselines for cumulative hard resets and idle time.

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 hyperparameter details for each algorithm, 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 code supplement also provides instructions for reproducing results.

Results: We plot results in Figure 2. 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 assign available humans but also when to *stop* requesting unnecessary supervision. C.U.R. also incurs fewer hard resets than baselines, especially Behavior Cloning, 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 cumulative idle time; C.U.R. idle time flattens out after the initial warmup.

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. See Figure 3 for the hardware setup. The objective of each robot is to reach a goal position is 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 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

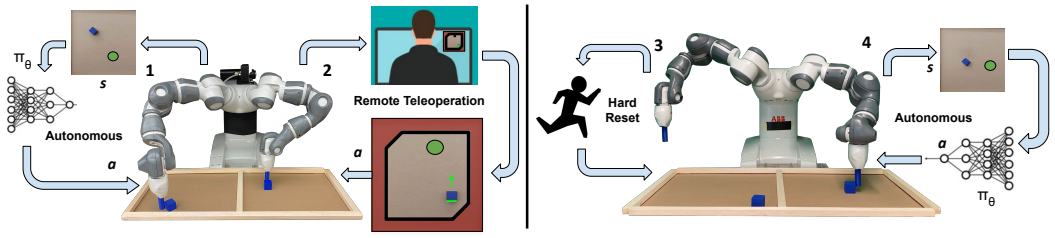

Figure 3: **Physical Task Setup:** an example timestep t in the physical experiment with 2 humans and 4 independent identical robot arms each executing the block pushing task. **Robot 1** queries robot policy π_{θ_t} for an action given an overhead image of the workspace and executes it in the workspace. **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.

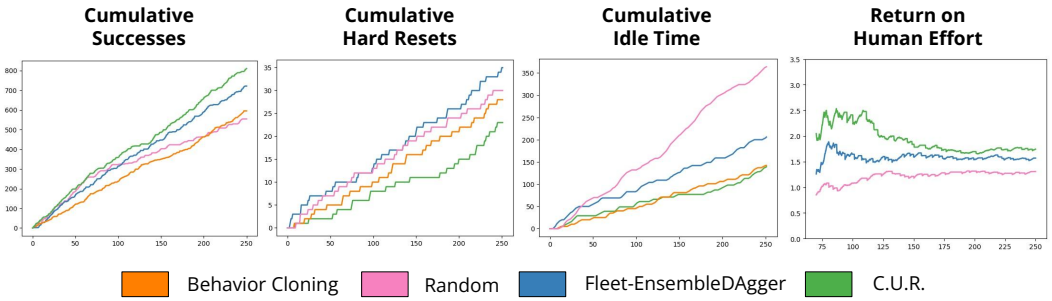

Figure 4: Physical results for the block-pushing task with 4 robots and 2 humans, where the x-axis is timesteps. C.U.R. achieves higher ROHE and cumulative successes as well as lower cumulative hard resets and idle time. However, ROHE does not noticeably improve, likely due to the low fleet operation time T.

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 robot actions per algorithm. All algorithms are initialized with an offline dataset of 5000 image-action pairs (500 teleoperated actions with $10\times$ data augmentation).

Results: We plot results in Figure 4. We observe that the C.U.R. algorithm achieves a higher ROHE, higher cumulative successes, lower hard resets, and lower idle time than baselines, albeit by a small margin. However, the ROHE for all algorithms remains flat over time instead of improving. Results suggest that (1) the fleet operation time of $T = 250$ is too short to observe improvement for a relatively high-performing initial policy π_{θ_0} and (2) U-prioritization in its current form is less suitable for real-world multimodal human supervisors than it is for deterministic algorithmic supervisors. 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 assumed to be homogeneous, (2) all robots operate in the same state space with the same distribution, (3) all robots are independent and we do not consider robot coordination, (4) humans have perfect situational awareness [22] and can move to different robots without any switching latency, and (5) we assume a constant hard reset time. 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 run more physical experiments to evaluate the applicability of the IFL benchmark results to real world tasks. We will also study reinforcement learning algorithms for IFL and extend the number of features and supported environments in the IFL benchmark suite. We hope that other robotics researchers will develop their own IFL algorithms and evaluate them using the benchmark toolkit to accelerate progress.

References

[1] 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/2p89zb35.

[2] Logistics Automation with Plus One Robotics. *Parcel Monitor*, May 2022. URL https://www.parcelmonitor.com/blog/tech-spotlight-logistics-automation-with-plus-one-automation/.

[3] 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.

[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] 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.

[6] B. Kim and J. Pineau. Maximum mean discrepancy imitation learning. In *Robotics: Science and systems*, 2013.

[7] 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.

[8] 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.

[9] 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.

[10] 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 *International Conference on Automation Sciences and Engineering (CASE)*, 2021.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] T. Ji, R. Dong, and K. Driggs-Campbell. Traversing supervisor problem: An approximately optimal approach to multi-robot assistance. *arXiv preprint arXiv:2205.01768*, 2022.

[32] 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.

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

[34] 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.

[35] M. Gombolay, R. Wilcox, and J. Shah. Fast scheduling of multi-robot teams with temporospatial constraints. 2013.

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

[37] A. Daw, R. C. Hampshire, and J. Pender. How to staff when customers arrive in batches. *arXiv e-prints*, pages arXiv–1907, 2019.

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

[39] 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.

[40] 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.

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

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

[43] 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.

[44] 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.

[45] 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.

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

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

[48] 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.

[49] 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.

[50] 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.

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

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

[53] 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.

[54] 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.

[55] 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.

[56] G. Kahn, P. Abbeel, and S. Levine. LaND: Learning to Navigate from Disengagements. In *arXiv preprint arXiv:2010.04689*, 2020.

[57] S. Vats, O. Kroemer, and M. Likhachev. Synergistic scheduling of learning and allocation of tasks in human-robot teams. *arXiv preprint arXiv:2203.07478*, 2022.

[58] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

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

[60] 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. *arXiv preprint arXiv:2203.15103*, 2022.

[61] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022.

[62] 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.

[63] 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.

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

[65] B. Thananjeyan*, A. Balakrishna*, S. Nair, M. Luo, K. Srinivasan, M. Hwang, and J. E. Gonzalez. Recovery rl: Safe reinforcement learning with learned recovery zones. 2020.

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

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