Breadcrumbs to the Goal: Supervised Goal Selection from Human in the loop Feedback

a dissertation presented
by
Marcel Torné Villasevil
to
Institute of Applied Computational Sciences,
School of Engineering and Applied Sciences

in partial fulfillment of the requirements
for the degree of
Master of Engineering
in the subject of
Computational Science and Engineering

Harvard University
Cambridge, Massachusetts
May 2023

Thesis advisor: Professor Pulkit Agrawal Marcel Torné Villasevil

Breadcrumbs to the Goal: Supervised Goal Selection from Human in the loop Feedback

Abstract

Effective exploration is a major challenge in robot learning due to large state spaces. Exploration is guided by relying on the generalization of the policy[18], failing due to exploration collapsing, uniform frontier expansion[29], causing over-exploration and hence being sample inefficient, and, directed frontier expansion, which needs signal to guide in the right direction. Prior work has shown the effectiveness of humans providing online feedback to direct exploration[21]. We propose a novel algorithm, that is more feedback efficient than prior work and does not require for constant online feedback for the policy to learn. The algorithm is called Frontier Expansion with Human Guidance (FewHug) and learns a goal selector from human feedback on pair-wise state comparisons to guide exploration during policy learning. We show the goal selector can still be trained using noisy human feedback and from a crowdsourced pool of 109 non-expert human annotators coming from different backgrounds. Finally, FewHug succeeds in playing the famous game of Bandu, consisting of assembling tower-like structures with blocks of different shapes, among other long horizon navigation and manipulation tasks in both the real world and simulation.

Contents

Listing of figures

To my family and friends who always believed in me,
to those who are giving me the opportunity to do something great.

Acknowledgments

Thank you to everyone who has supported me along this journey. First of all, I want to thank my parents for the education and all of the amazing opportunities they have provided me with. I want to thank my parents and my sister for their inconditional support in my pursuits. My friends have been an essential part of my journey so far, keeping me company and support throughout. I also want to thank Prof. Abhishek Gupta for his mentorship, supervision, and for teaching me so much during this last year. I also want to thank Prof. Pulkit Agrawal for his supervision and for giving me the opportunity to be a part of his lab, which is the reason why I could work on this great project I am so proud of. I want to thank my thesis committee, Prof. Pulkit Agrawal, Prof. Susan Murphy, and Dr. Weiwei Pan. Finally, I want to thank my collaborators, my supervisors, Max Balsells i Pamies (University of Washington), Tao Chen (MIT), and Zihan Wang (University of Washington), who have helped me bring this challenging project to light.

0

Introduction

How should we teach agents a new task? A general method is to provide agents with a reward function and employ reinforcement learning. However, not only does learning from rewards consume large amounts of data, but in practice, the reward function often needs to be iteratively modified by the human designer to avoid reward hacking or overcome exploration challenges. The loop of humans designing a reward function, observing the behavior of learned policy, and then re-designing rewards to improve performance is tedious and data inefficient. To circumvent this problem, prior work has explored different methods for learning from human feedback, such as preference-based learning[12], learning reward functions from language, learning from demonstrations, etc. However, the prominent challenges are that humans can provide noisy feedback, may not know the optimal solution to the task at hand and some approaches require large amounts of human feedback (e.g., DAGGER[35]).

If the agent was better at autonomously exploring its environment, it could learn from a sparse reward function that is easier to design and thus be less reliant on humans designing and iterating over detailed reward functions providing dense reward supervision. Several exploration strategies exist that incentivize the agent to visit novel states and have led to impressive performance gains in sparse reward scenarios. These strategies function by either expanding the frontier of states visited by the agent or explicitly reward the agent for visiting novel-states. As such it is possible that while maximizing novelty, the agent can get lost: blindly exploring new states may not aid the agent in solving the desired task. E.g., suppose one is visiting a new city, but doesn't has access to a city map and needs to get to a landmark. If the person explores by walking on all roads of a city, it can take forever to reach the landmark. On the other hand, if the person got guidance from a city dweller, she could explore in a directed manner and quickly reach the destination.

Our desiderata is a method for learning complex long-horizon tasks that is not overly reliant on human feedback and is robust to noise in human supervision. Instead of solely relying on human feedback or novelty-seeking exploration, a methodology that efficiently leverages both these learning mechanisms can mitigate the drawbacks of each individual method. Superior exploration can reduce the need for human feedback and on the other hand, human feedback can prevent an exploring agent to get lost. The question, therefore, is how should human feedback and an agent's self-exploration be combined?

To be more precise, we require our system to meet four criteria.

- Minimal human feedback (**minimal**)

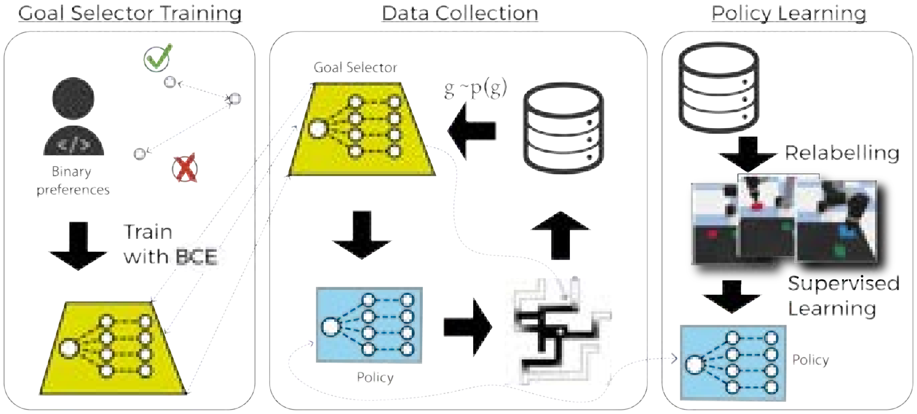


Figure 1: Overview of FewHug. Our method consists of three parts. **left**: a goal selector is trained querying a human annotator for binary preferences on achieved states. **middle**: during trajectory rollouts for data collection, the goal selector is used to guide exploration. **right**: the policy is learned using hindsight relabelling on the collected rollouts.

- Feedback should not need to be provided online, but asynchronously instead (**asynchronous**)
- The system should be robust to noisy and multimodal human feedback (**noisy**)

Conditions **minima** and *asynchronous* are there to reduce the burden on the supervisor. Collecting large scale crowdsourced data is challenging since human preferences can be noisy and multimodal, due to difference in preferences. Hence the importance of the **noisy** condition. Satisfying the 3 conditions allowed us to train an agent to solve a long-horizon task collaboratively, among 109 annotators across the globe, with less than two minutes of labelling per person.

Our proposed algorithm, Frontier Expansion with Human Guidance (FewHug), leverages human feedback to guide the directed selection of *which* state to start exploring from. This encourages exploration in the directions that are *important* while ignoring irrelevant directions. FewHug solicits binary preference comparisons between *visited* states, and asks the human annotator for which one of the two is closer to the goal. To make more efficient use of the human feedback (**minimal**), we use the annotations not to directly select the goal[21] but instead to train an approximate model of state-goal distances, the goal selector. The goal selector is used to bias goal sampling during explo-

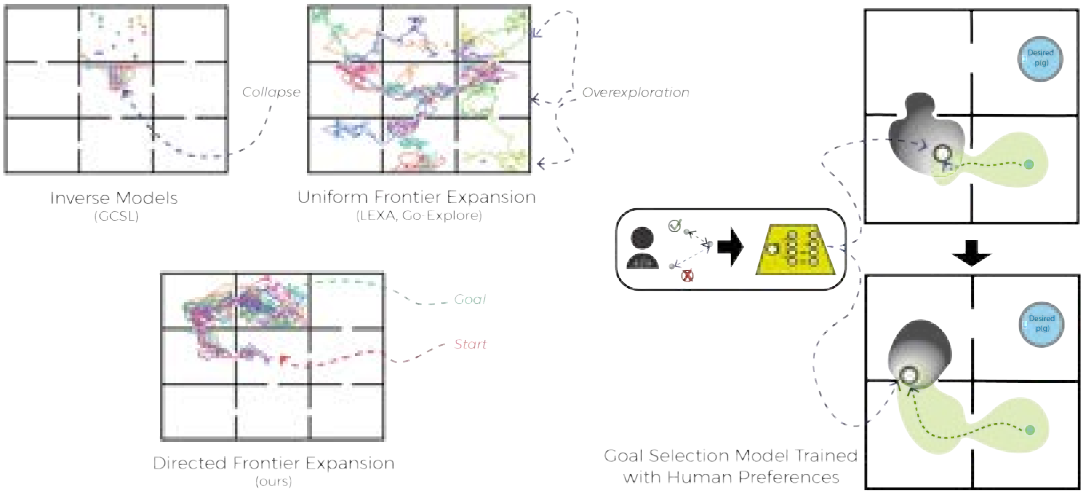


Figure 2: left: Comparison of exploration algorithms for goal-reaching, highlighting the benefits of directed frontier expansion over inverse models and uniform frontier expansion. Aerial views of floor plans with 9 rooms and multiple trajectories are shown. **right**: The schematic on the right provides an overview of FewHug in a four room benchmark, demonstrating the direct expansion of the frontier through human-guided goal selection. The area in green consists of the visited states, the area in black consists of the explored frontier.

ration by selecting with higher probability the closest *visited* states to the goals as *exploration* goals. The policy explores by revisiting these *achieved* states and subsequently expanding the frontier of visited states towards the true goal. The data collected during this exploration is relabeled in hindsight and used for self-supervised goal-conditioned policy learning [18] without requiring the human supervisor to be present. This allows humans to provide feedback asynchronously and infrequently, with the agent relying on self-supervision for a bulk of policy learning (**asynchronous**). By providing these comparisons, humans are able to essentially "drop breadcrumbs", incrementally guiding agents to reach distant goals, as shown in Fig 3.1. Finally, we note that compared to previous RLHF methods[11] where the human feedback is converted into a reward function and has a direct effect on the policy update, in FewHug the human feedback has *only* an effect on the direction of frontier expansion. In the presence of noisy human feedback this will degrade the effectiveness of the exploration. However, since the the policy learning relies on self-supervision its update remains disentangled from the human feedback and makes the policy learning more robust to noisy and misspecified human feedback (**noisy**).

Overall, this work presents the following contributions:

Guided Frontier Expansion from minimal Human Feedback: We guide exploration in policy learning using a parametric goal selector learned from human preferences, which is more efficient use of human feedback than using this directly.

Method suited for large-scale crowdsourced human-feedback data collection for policy learning: We show FewHug scales to collecting data from 109 non-expert annotators across the world.

Method suitable for real-world policy learning: We show FewHug can learn policies in the real world because of its effective frontier expansion, the sample efficiency of the policy learning algorithm and its compatibility with pretraining from trajectory demonstrations.

1
Related Work

Our work is at the intersection of three different fields – exploration in reinforcement learning, goal-conditioned reinforcement learning, and reinforcement learning from human preferences. Next, we will give an overview of each one of these subfields with the drawbacks of current work in each area and how FewHug either builds or improves upon concepts from the mentioned fields.

1.1 Exploration in Reinforcement Learning

Our work can naturally be connected to the problem of exploration in RL. While exploration is a widely studied subfield in reinforcement learning [8,33,4,29,7,15], this is typically concerned with either the exploration-exploitation tradeoff [3] or maximizing state coverage [4,8,29], as opposed to our goal of performing *targeted* exploration informed by human feedback. Particularly related to our work is Go-Explore [15], a paradigm for exploration that aims to maintain a "frontier" for state expansion, that gradually expands to maximize state coverage. However, as seen in Figure 2, novelty-based exploration methods suffer from *overexploration* making it sample inefficient. Moreover, self-supervision techniques such as hindsight relabeling [1,18,28] have been proposed to get around the challenge of exploration for these problems. By learning goal-reaching behavior in *hindsight* for goals that were actually reached, these methods obtain dense supervision, which can aid with exploration through policy generalization across goals. These methods on the other side, suffer from *underexploration* due to the policy collapsing. Hence, the failures of prior work by *overexploration* and *underexploration* suggest we need a way to encourage self-supervised policy learning techniques to explore in *directed* ways, as shown in 2.

1.2 Goal-Conditioned Reinforcement Learning

Goal-conditioned RL algorithms[26,18,28,1,25,Kaelbling] are multi-task RL methods where various tasks are defined as reaching different goal states. A number of different approaches have been proposed for goal-conditioned RL - learning with hindsight relabeling [1,14,25,Kaelbling], learning latent spaces for reward assignment [32,34], learning dynamical distances [21,16] and goal conditioned imitation learning [19,28,18,31]. While these algorithms are able to solve tasks with simple exploration, they can struggle with tasks with complex sequential nature. In contrast, our work shows the ability to solve sequential goal-reaching problems, just using binary human-in-the-loop comparisons for guidance.

Most closely related to our work is the paradigm introduced in Hartikainen et al.[21]. In Hartikainen et al.[21], a human supervisor occasionally selects states to explore, which, when combined with self-supervised Q-learning, can solve goal-conditioned RL problems. However, this brings an inefficient usage of human feedback, we show learning a parametric goal selector reduces the amount of human labels needed by 58% 3.6. Moreover, in this line of work exploration is limited to the particular state selected by the human, whereas in FewHug, the exploration frontier can continue expanding using the generalization of the learned goal-selection model 3.8.

1.3 Reinforcement Learning from Human Preferences (RLHF)

RLHF represents a significant breakthrough in the field of language models by substantially improving policy alignment compared to supervised methods[?]. In the robotics literature, human in the loop RL is a well-explored concept, with various interfaces ranging from preferences [11,5,6,24] to scalar rewards [23], language based corrections [37], binary right/wrong signals [10] and even sketching out rewards [9]. Many of these techniques learn how to ground human interfaces into a scalar reward signal that can then be used to guide RL in the single task setting [11,5,6]. A major focus of several of the prior works [6,5] has been on how to actively propose points to be labeled, reducing the burden on human supervisors. In this work, we take a complementary view and show that by considering human feedback in a self-supervised policy learning setting, the human effort is naturally reduced since much of the burden can be placed on self-supervision. Moreover, the self-supervision allows FewHug to show increased robustness to misspecified human feedback.

2

Frontier Expansion with Human Guidance

We propose a new algorithm that leverages human feedback to guide frontier expansion in policy learning, called FewHug: Frontier Expansion with Human Guidance. Next, we will set up the framework in which we are working, giving the necessary definitions and defining the mathematical symbols that we use. Finally, we will present FewHug in detail including each module that form this.

2.1 Problem Setup and Preliminaries

The work solves goal-reaching tasks by using goal-conditioned policy learning methods. The goal-reaching problem is characterized by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \rho(s_0), T, p(g) \rangle$, adopting the standard MDP notation with $p(g)$ being the distribution over goal states $g \in \mathcal{S}$ that the agent is tasked with reaching. We assume sampling access to $p(g)$. We aim to find a stationary goal-conditioned policy $\pi(\cdot|s, g)$: $\mathcal{S} \times \mathcal{S} \rightarrow \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ is the probability simplex over the action space. We will say that a goal is achieved if the agent has reached the goal at the end of the episode (or is within ε distance of the goal). The learning problem can be characterized as that of learning a goal-conditioned policy that maximizes the likelihood of reaching the goal $\pi \leftarrow \arg\max_{\pi} J(\pi) = \mathbb{E}_{g \sim p(g)} \left[P_{\pi_g} \left(s_T = g \right) \right]$.

Self-Supervised Goal-conditioned Policy Learning: Goal-conditioned Reinforcement Learning methods approach this problem using the reward function $r_g(s) = 1(s = g)$, defining a sparse reward problem: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi(\cdot|s,g), g \sim p(g)} \left[\sum_{t=1}^{H} \gamma^t r_g(s_t) \right]$. This problem can be difficult to solve with typical reinforcement learning algorithms[20,36,17] because the training process is largely devoid of learning signals.

To circumvent this challenge, we can exploit the structure of the goal-reaching problem using the hindsight relabeling technique [Kaelbling,2]. Hindsight relabeling leverages the insight that transitions that may be suboptimal for reaching "commanded" goals g, may be optimal in *hindsight* had the actually reached states s_T been chosen as goals. This allows us to relabel a transition tuple $(s, a, s', g, r_g(s))$, with a hindsight tuple $(s, a, s', g', r'_g(s))$, where g' can be arbitrary goals chosen in hindsight. When g' is chosen to be states s actually visited along a trajectory, the reward function $r_s(s) = 1$ provides a dense reward signal for reaching different goals $g = s$, which can be used to supervise an off-policy RL algorithm [2].

Several prior works [18] circumvent RL, instead directly leveraging supervised learning techniques for goal-conditioned policy learning. In particular, by repeatedly applying the principle of hindsight relabeling, suboptimal trajectory data can be converted into optimal data for supervised learning. Given a sampled trajectory $(s_0, a_0, \ldots, s_H, a_H)$ obtained by executing the policy $\pi(\cdot|s, g)$ for a commanded goal $g \sim p(g)$, hindsight relabeling allows for the conversion of a suboptimal dataset for goals $g \sim p(g)$ into an optimal dataset for goals $g \sim p_\pi(s_H)$ that were reached by the policy (2.2). This relabeled optimal dataset can then be used for supervised learning:

$$J_{\text{GCSL}}(\pi) = \mathbb{E}_{\tau \sim \mathbb{E}_g[\pi_{\text{old}}(\cdot|g)]} \left[\sum_{t=0}^{T} \log \pi(a_t | s_t, \mathcal{G}(\tau)) \right] \tag{2.1}$$

This process can be repeated, iterating between collecting data, relabeling it, and performing supervised learning.

$$\mathcal{D}_\tau = \{(s_t, a_t, g = s_{t+h}, h) : t, h > 0, t + h \leq T\} \tag{2.2}$$

2.2 Guiding Exploration in Goal-Conditioned RL with Human Feedback

In this work, we pose that it is relatively straightforward for human supervisors to guide an agent throughout its learning by indicating which achieved states bring it closer to the goal as if we dropped “breadcrumbs” to guide exploration of the agent. Doing so guides expansion of the frontier of visited states in *particular* directions, rather than simply encouraging indiscriminate frontier expansion. While several methods have explored how to leverage human feedback to guide exploration [12,5], the bottleneck is the amount and quality of human supervision required. We show that in human-in-the-loop policy learning, leveraging self-supervision via hindsight relabeling substantially alleviates the burden on human supervisors in terms of frequency and quality of feedback needed. In our proposed framework, FewHug, human feedback is used to bias which goal is commanded during exploratory data collection while relying on self-supervised hindsight relabeling to learn from this collected data. This allows for learning goal-reaching policies in an unbiased way while leverag-

Algorithm 1 FewHug: Frontier Expansion with Human Guidance for Goal-Conditioned Policy Learning

```
Input: Human/Computational oracle $\mathcal{H}$, goal distribution $p(g)$
Initialize policy $\pi$, goal selection model $\mathcal{G}$, data buffer $\mathcal{D}$
while True do
    Sample goal $g \sim p(g)$
    $\mathcal{D}_\tau \leftarrow$ RolloutPolicyWithExploration($\pi, \mathcal{G}$, g, $\mathcal{D}$)
    $\mathcal{D}_{\tau'} \leftarrow$ RelabelTrajectory($\mathcal{D}_\tau$) (Equation 2.1)
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}'_\tau$ {Add relabelled trajectory to the buffer}
    $\pi \leftarrow$ TrainPolicy() (Equation 2.1)
    Collect preference feedback from $\mathcal{H}$ (Section 2.2.1)
    $\mathcal{G} \leftarrow$ TrainGoalModel($\mathcal{D}, \mathcal{H}$) (Equation 2.3)
end while
```

ing human supervision to guide exploration. FewHug learns goal-reaching policies by alternating two phases - (1) exploration for data collection, informed by human feedback, and (2) hindsight relabeled learning for policies, which we describe next.

2.2.1 Exploration: Guided Frontier Expansion for Data Collection via Binary Comparisons

Suppose the target goal distribution is $p(g)$. Goal-reaching algorithms typically learn a policy using the data generated as follows: randomly sampling a goal $g \sim p(g)$, and then collecting data by sampling actions from a stochastic goal-conditioned policy $\pi(\cdot|s, g)$. As discussed in Figure 2 and in more detail in Appendix C.0.1, such a random goal sampling strategy often collapses to trivial solutions or leads to lousy exploration. A natural solution to ensure exploration in this setting is to do frontier expansion [15] - maintain a frontier of states $\mathcal{F}$ that have been visited by the policy during data collection and gradually expand the frontier by revisiting particular "states of interest" on the frontier and performing expansion starting from these states. This has been shown to be an effective

Algorithm 2 RolloutPolicyWithExploration

```
**Input:** Target goal $g$, policy $\pi$, goal selection model $\mathcal{G}$, data buffer $\mathcal{D}$
$\mathcal{D} \leftarrow \{\}$
$g_b \sim \text{softmax}(\mathcal{G}(s, g); \forall s \in \mathcal{D})$ {Select breadcrumb goal }
**for** $i = 1, 2, \ldots, N$ **do**
    Sample initial state $s_0$
    **while** NOT stopped **do**
        Sample action $a \sim \pi(a|s, g_b)$, execute in environment
    **end while**
    Execute $\pi_{\text{random}}$ until $H$ timesteps for exploration
    Add $\tau$ to $\mathcal{D}'_\tau$ without redundant states
**end for**
```

tool [15] for indiscriminate exploration in reinforcement learning by simply choosing the "states of interest" that have the lowest visitation counts in the frontier. In this work, we instead leverage binary comparison feedback provided by a human supervisor to select which states to expand the frontier from, providing more targeted exploration than novelty-based frontier expansion.

Frontier Expansion with GCSL: We build on the framework of goal-conditioned supervised learning (Section 2.1) for self-supervised policy learning. In this case, the frontier $\mathcal{F}$ can simply be the set of all states that have actually been visited thus far by the goal-conditioned policy (i.e $\mathcal{F} = \mathcal{D}$, and we will use them interchangeably from here on), and revisiting a particular breadcrumb state $g_b \in \mathcal{F}$ can be done by executing the policy $\pi(a|s, g_b)$ in the environment sequentially until the desired goal g_b is reached. This can be accomplished with high probability for states that have already been seen since the policy is trained via hindsight-relabeled supervised learning [18]. Frontier expansion is performed by executing random exploration from the reached state g_b, and adding the collected data to the buffers, as shown in Fig 2. This process can then be repeated to continue exploring to collect data and train the policy to reach further states.

Learning State-Goal Distances from Binary Comparisons: We propose a simple interface between the human and the algorithm, we simply rely on the human supervisor to provide binary comparisons of which state-goal pairs are closer than others. As we will describe next, these comparisons can be used to learn a parametric *model* of distances between states and goals. This model can then be used for the frontier goal selection in Algorithm 1, even in the absence of a human constantly supervising the system.

Every K episodes, we display a human labeler a triplet of goal g, a first comparison state s_1, and a second comparison state s_2. The human labeler must provide a comparison of which state amongst $\{s_1, s_2\}$ is *closer* to the goal g. As studied in a number of prior works [11,6,5], these "binary comparisons" can be used to train an unnormalized estimate of distances $f_\theta(s, g)$ by leveraging the Luce-Shepard rule[27]:

$$\begin{aligned} \max_\theta \sum 1(s_1 > s_2|g) \log \frac{\exp f_\theta(s_1, g)}{\exp f_\theta(s_1, g) + \exp f_\theta(s_2, g)} + \\ (1 - 1(s_1 > s_2|g)) \log \frac{\exp f_\theta(s_2, g)}{\exp f_\theta(s_1, g) + \exp f_\theta(s_2, g)} \end{aligned} \tag{2.3}$$

This objective encourages states s closer to particular goals g to have smaller $f_\theta(s, g)$.

Using Learned State-Goal Distances for Directed Frontier Expansion: The unnormalized estimate of distance $f_\theta(s, g)$ can be used to select which breadcrumbs g_b to command for frontier expansion during exploration, by sampling breadcrumbs in proportion to their negated exponential distance to goal, $\exp(-f_\theta(s, g))$. Intuitively this encourages greater frontier expansion towards states that have a lower estimated distance to the goal since these are more promising directions to explore in, as indicated by the human-provided comparisons.

$$g_b \sim p(g_b|g); p(g_b|g) = \frac{\exp f_\theta(g_b, g)}{\sum_{g' \in \mathcal{D}} \exp f_\theta(g', g)} \tag{2.4}$$

where $\mathcal{D}$ represents the set of all reached goals. This softmax sampling ensures that local errors in the reward model are smoothed out, making the algorithm resilient to misspecification. Once the commanded state g_b is reached, the frontier can then be expanded by subsequently performing several steps of random sampling from an exploratory distribution such as the uniform distribution $\mathcal{U}$.

It is important to note here that the parametric reward model can ensure continued expansion of the frontier even when a human supervisor is not present. This reduces the burden on a human supervisor since they can provide comparisons asynchronously and infrequently rather than constantly being in the loop as is common in prior work [11,10].

2.2.2 Policy Learning: Hindsight Relabeled Learning for Goal Conditioned Policies

Given the exploratory data collected by guided frontier expansion in Section 2.2.1, we can leverage a simple supervised learning scheme, building on [18] for goal-conditioned policy learning. Let us assume access to trajectories $\tau = \{s_0, a_0, s_1, s_2, \ldots, s_T, a_T\}_{i=1}^{N}$ obtained from guided frontier expansion (Algorithm 2, Equation 2.4). Given these potentially "sub-optimal" trajectories, we can perform self-supervised hindsight relabeling as in [18] to construct a dataset of optimal tuples Eq. 2.2.

$\mathcal{D}_\tau$ can be treated as optimal. These can then simply be used for supervised learning of policy $\pi(\cdot|s, g)$ with standard maximum likelihood learning $\max_\pi \mathbb{E}_{(s,a,g)\sim\mathcal{D}_\tau} [\log \pi(a|s, g)]$.

As policy learning continues improving, the learned policy can be deployed to solve an expanding set of goals, eventually encompassing the desired goal distribution.

The important thing to note here is that since the policy learning is completely self-supervised, FewHug can continue learning meaningful behavior even in the absence of constant human supervision and even in the presence of *misspecified* human supervision, making it far more resilient and practical for use in human-in-the-loop reinforcement learning.

The overall pseudocode of FewHug is shown in Algorithm 1 and we refer the reader to the Appendix for full consideration of practical implementation details.

3

Results of the Frontier Expansion with Human Guidance Algorithm

In this work, we show that FewHug can learn to successfully achieve long-horizon tasks, and tasks with large combinatorial exploration spaces through little human supervision. To demonstrate these experimentally, we test on several goal-reaching domains in simulation in the MuJoCo[38] and PyBullet[13] simulators as well as real-robot experiments and experiments with both synthetic and real human annotations.

3.1 Evaluation Domains

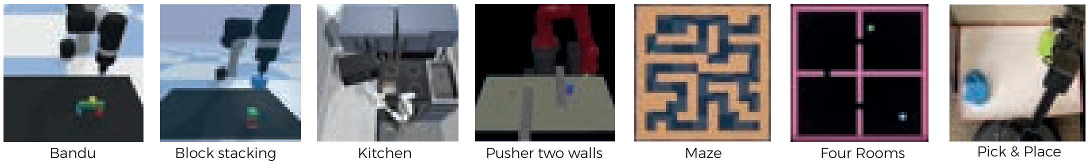


Figure 3.1: We test our method on six varied benchmarks in simulation and one in the real world. **Four rooms** and **Maze** consist of 2D navigation task. **Pusher with two walls**, **Kitchen**, **Block Stacking**, and **Bandu** are manipulation tasks, and **Pick & Place** is in the real world.(for more details please read Appendix A).

We evaluate FewHug on a variety of benchmarks presented in Figure 3.1. For further details on the domains, such as their computational oracle distance models and more, we refer the reader to Appendix A.

3.2 Baseline Comparisons and Evaluation Metrics

We compare FewHug to relevant baselines from prior work. These baselines are chosen to compare FewHug with methods that perform pure exploration, hindsight relabeling, and human preferences without being goal conditioned to highlight the benefits of combining goal-driven self-supervision with human-in-the-loop exploration guidance.

1. **GCSL:** We compare with the iterative supervised learning algorithm for goal-reaching introduced in [18], consisting of hindsight relabeling without additional exploration.

2. **Learning from Human Preferences:** We consider the technique introduced in [11], which learns a goal-agnostic reward model using binary cross-entropy. This learned reward is then combined with an on-policy RL algorithm[36] to learn the policy.

3. **DDL:** Dynamical Distance Learning[21] proposes a method to learn a goal-conditioned reward function by regressing on the time distance between states achieved in the same trajectory. A human synchronously provides preferences on which state brings the agent closest to the goal, note that no goal selector is being learned. The policy is then trained to maximize the learned reward to get to this selected state.

4. **Go-Explore/LEXA:** We compared with a version of goal-reaching with indiscriminate exploration. In particular, we perform frontier goal selection by identifying goals with the lowest densities. The policy returns to these states and perform random exploration from there. This is equivalent to performing indiscriminate exploration.

5. **Proximal Policy Optimization:** We compare with an on-policy algorithm [36] with both a standard sparse and dense reward to directly optimize the goal-reaching objective.

6. **Behavior Cloning**: Supervised learning on a batch of expert trajectories. In our experiments we use 5 expert trajectories.

7. **Behavior Cloning + Ours**: We pretrain the policy using imitation learning and we warm start our goal selector by training it from the expert trajectories. Given two random states in the same expert trajectory we add them into the training data for the goal selector, setting the state further in time as closest to the goal.

3.3 Learning Goal-Conditioned Policies with Synthetic Human-in-the-Loop Feedback in Simulation

We consider goal-reaching problems in the six simulation domains shown in Fig 3.1. These are domains with non-trivial exploration challenges —the agents must navigate around walls, sequence multiple behaviors and purely random exploration is unlikely to succeed. We evaluate FewHug

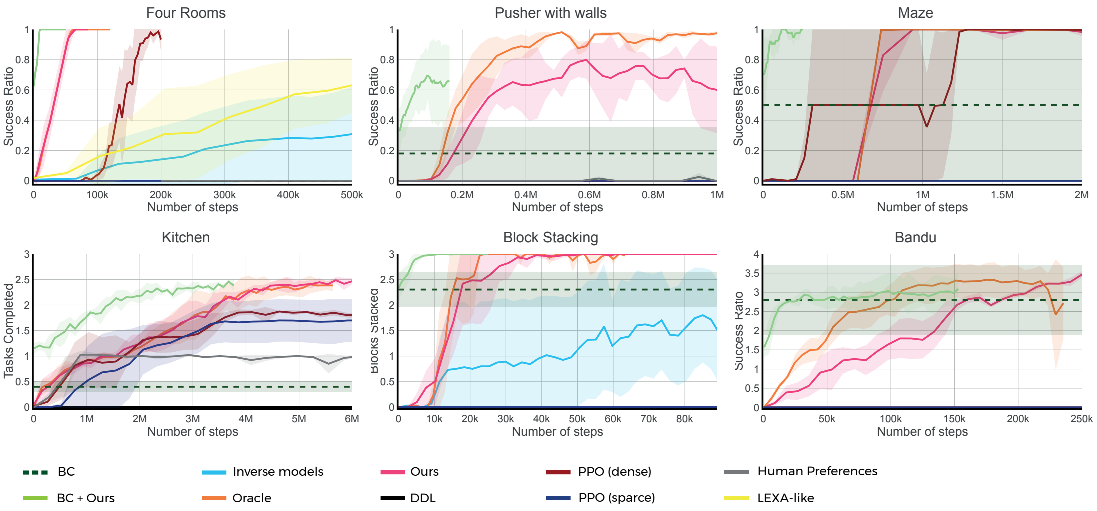
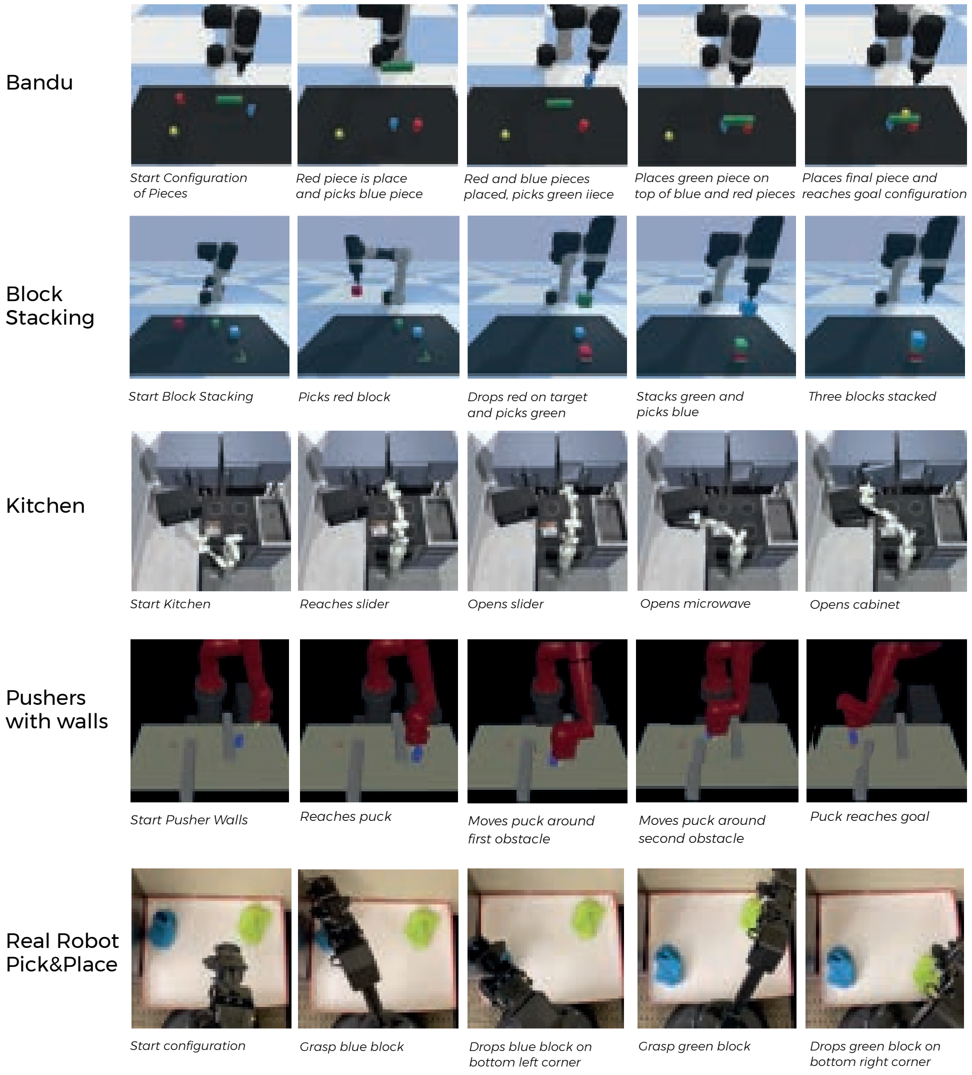


Figure 3.2: Train success of FewHug on the proposed benchmarks compared to the baselines. We show our method outperforms the rest of the baselines some of which cannot solve the environment, converging to the oracle accuracy. Note the lexa-like benchmark is only computed in the four rooms benchmark.

compared to the baselines described in Section 3.2 on these domains . We report the percentage of goals reached successfully in Fig 3.2 as learning progresses.

It is clear from Fig 3.2, that guiding exploration using human feedback is *significantly* better than techniques that perform purely exploration [15,29] or purely hindsight relabeling [18,1], on-policy reinforcement learning [36] and even (non-goal conditioned) learning from human preferences [11]. FewHug performs directed frontier expansion, which in high dimensional state spaces will explore much more efficiently than indiscriminate exploration methods like LEXA. On the other hand, methods like GCSL and HER do not explicitly expand the frontier but instead purely rely on policy generalization and stochasticity to perform exploration. This can fail in long-horizon task domains like the ones in Fig 3.1. Goal-conditioned reinforcement learning methods with PPO do not experience enough reward signals to actually learn directed behavior. And lastly, learning from human preferences without any goal conditioning struggles with learning complex behaviors using PPO and a non-stationary reward function.

The *oracle* line corresponds to our method without learning a goal selector, meaning that the closest achieved goal is always being commanded. The *ours* line corresponds to learning the goal selector to select the goal to expand. From Fig 3.2, we see that the learned model performs slightly worse than the oracle in some domains, but is largely comparable in most domains.


Bandu
Start Configuration of Pieces
Red piece is place and picks blue piece
Red and blue pieces placed, picks green iiece
Places green piece on top of blue and red pieces
Places final piece and reaches goal configuration
Block Stacking
Start Block Stacking
Picks red block
Drops red on target and picks green
Stacks green and picks blue
Three blocks stacked
Kitchen
Start Kitchen
Reaches slider
Opens slider
Opens microwave
Opens cabinet
Pushers with walls
Start Pusher Walls
Reaches puck
Moves puck around first obstacle
Moves puck around second obstacle
Puck reaches goal
Real Robot Pick&Place
Start configuration
Grasp blue block
Drops blue block on bottom left corner
Grasp green block
Drops green block on bottom right corner

3.4 Learning Goal Conditioned Policies with Real Human Annotators

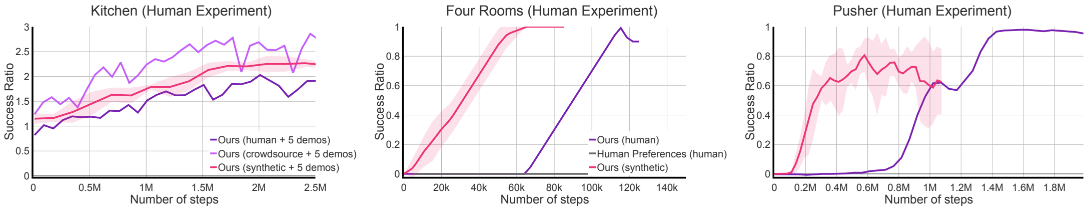


Figure 3.3: Learning Progress with Human-in-the-Loop Feedback for the kitchen (**left**), four rooms navigation (**middle**) and pusher walls (**right**) for which we collected 1600, 660 and 2780 labels respectively.

The previous experiments were done using synthetic human annotations without any added noise. However, the goal of our work is to show that we can learn these goal selectors from real human feedback, that can be noisy and multimodal. We start with experiments on three of the simulations tasks: the kitchen, pusher and four rooms, with 2 to 6 annotators per experiment. In Figure E.2, we show how we succeed to learn successful policies from scratch on both the pusher and the four rooms experiment (*Ours (human) line*). We collected 660 labels for the four rooms environment from 2 different annotators and 2780 labels for the pusher from 4 different annotators. Furthermore, we tried training a policy on the four rooms environment with the same number of labels as for FewHug, however, as we see it did not work. Furthermore, on the kitchen we pretrained the policy with some demos to warm start and train our policy faster. We successfully trained a policy from 6 annotators and a total of 1600 labels. These results show how FewHug can successfully working from real human preferences.

FewHug is capable to do even more, for this reason, we ran a crowd-sourcing experiment using a platform that we designed from scratch specifically for this project, but which could be reused for any algorithms requesting human preferences, we refer the reader Fig E.1 to see the interface. We ran the experiment during 12 hours with data from 109 annotators across the world, as seen in 3.4. In table 3.1 and E.1, we provide more details about the demographics of our annotators. We spanned the around all over the world, having annotators leaving in 13 different countries, with ages ranging from 18 to 65+ and a variety of academic backgrounds. Each annotator could provide labels at any time during the day, our recommendation was to provide 30 labels, which took on average less than 2 minutes of their time. We collected a total of 2678 labels for this crowd-sourcing experiment.

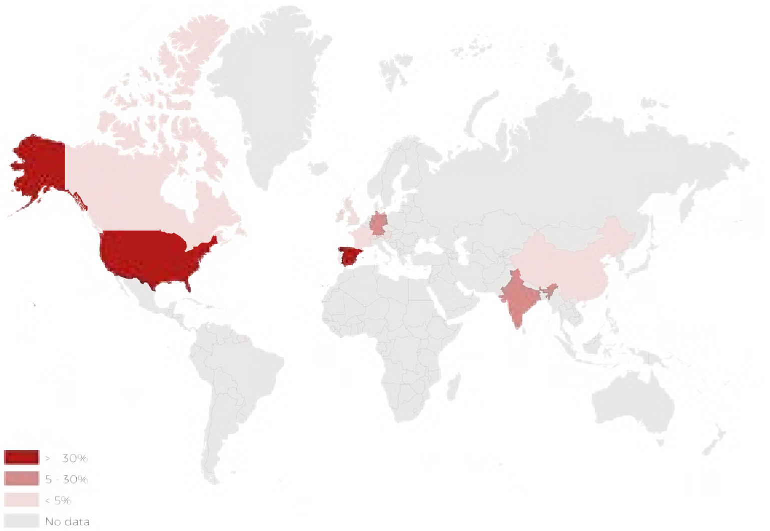


Figure 3.4: World map depicting the country of residence of our crowdsourcing experiment. We had a total of 109 annotators currently living in 13 different countries. See 3.1 for further details.

3.5 Learning Policies in the real world

FewHug's qualities of being robust to **noisy** feedback and requiring **minimal** and asynchronous human supervision together with its self-supervised policy learning nature and the capability to be pretrained from trajectory demonstrations makes it suitable for learning in the real world. Hence, we show this in practice, learning a policy for pick and place with the LoCoBot hardware[30]. Firstly, the state space becomes image space, instead of point space as was used in the previous experiment. In Figure 3.5, we show FewHug is compatible with working from image space in two of the simulation tasks, the four room navigation and the block stacking. In Figure 3.5, we also show the success curve when learning the task in the real world. The setup consisted of images as state space, the action space was the continuous x,y position where we had to execute the grasp. We collected 132 labels and the robot was trained for around 30 hours.

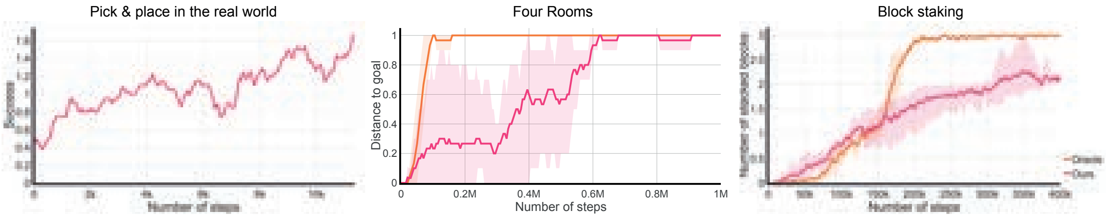


Figure 3.5: Success rate curves for the real robot pick and place experiment (**left**) and the simulated benchmarks: four rooms (**middle**) and pick and place (**right**) all using images as state space, succeeding in all. Both goal selector and policies were taking images as state space. For dealing with the stopping criteria in image spaces we use an image classifier trained using contrastive learning to determine which images represent close states.

3.6 Ablation Analysis

Lastly, we conducted a number of quantitative analysis experiments to show further benefits of FewHug.

Learning a goal selector is more feedback efficient than directly using the human feedback In figure 3.6 (**top-left**) we show a comparison of the number of labels needed to succeed when using a parametric goal selector (Ours) against directly using the goal selected by the human (DDL). We show the comparison between different frequencies of human querying. 15, 100, 500 episodes are the number of episodes we wait before querying the human annotator for more labels. We observe that when learning a goal selector, we obtain a reduction in the number of labels needed of 40% when querying every 15 or 100 episodes and a reduction of 59% when querying every 500 episodes. Furthermore, if we don't learn this parametric model, with low fre-

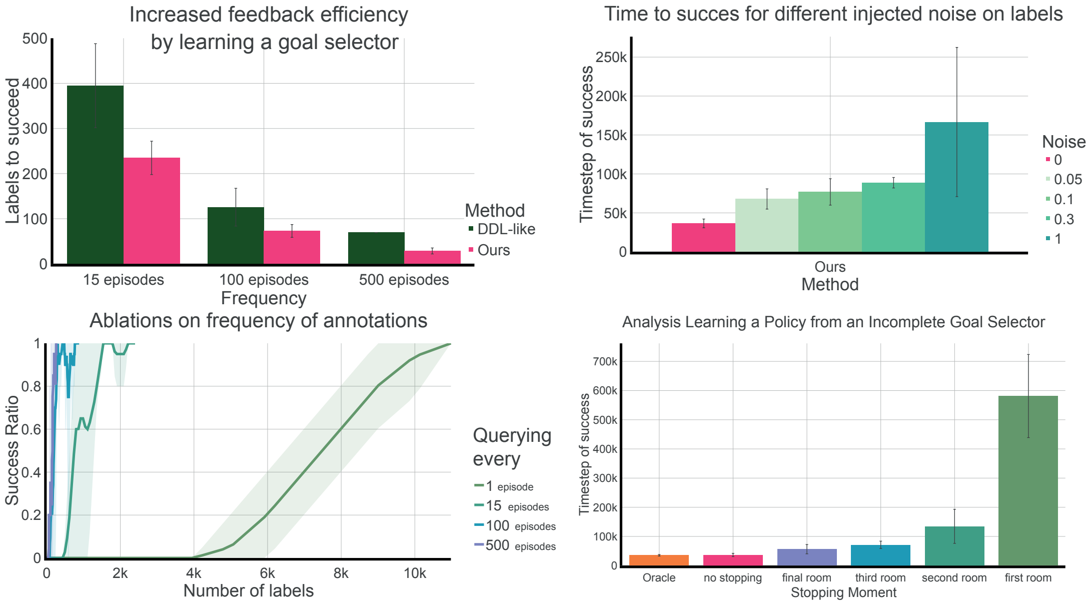


Figure 3.6: Analysis of FewHug on the four rooms benchmark. **(top-left)**: Comparison in the human feedback needed when learning a goal selector (Ours) or not (DDL), we see learning a goal selector is more efficient by 40% when querying for feedback every 15 and 100 episodes, and 59% when querying every 500 episodes. **(top-right)**: Effect of adding Gaussian noise in the labels provided by the human. FewHug is robust to different amounts of added noise, however, as noise increases, so will the timesteps needed to succeed. Noise is injected into the distance function used by the synthetic human to provide labels. For reference, the distance between the initial state and the goal is around 1.6. **(bottom-left)**: Effect on the number of labels needed for a given querying frequency, lower frequency is more feedback efficient, but comes at a cost of sample complexity as seen in Figure D.3. **(bottom-right)**: Freezing the goal selector at different points in the learning of the policy and how long it takes to learn a successful policy. We see that an earlier stop in the training leads to an increase in the timesteps needed to succeed. FewHug is robust it is against incomplete goal selectors but takes longer to find the solution if we stop earlier.

quencies we might not learn a successful policy, as happens for the non-parametric version at 100, 500 episodes of frequency. When using the non-parametric goal selector (DDL) not all trials succeed, for querying every 100 episodes, 2 seeds out of 4 fail and for 500 episodes between querying 3 out of the 4 fail, which is another reason why parametric goal selectors are better.

FewHug is robust to noisy labels Increasing the noise in human labels leads to an increase in the number of timesteps needed to for the policy to learn to achieve the goal, as seen in Figure 3.6. However, this does not decrease the accuracy of the resulting policy. Increased noise in the labels makes exploration become less directed and closer to the uniform frontier expansion methods.

Having a closer look in Figure 3.7 at the shape of the reward function when large noise is added to the feedback. We observe that the goal selector becomes less accurate compared to the one with perfect feedback in Fig 3.8. However, FewHug still successfully reaches the goal. As we can see, there are 3 modes in the final step (4th subfigure in 3.7). This means, the goals will be sampled most frequently from these 3 modes, which will result in a less efficient frontier expansion, since only one of the three modes is the target goal. However, since we are learning a goal-conditioned policy through self-supervised learning this remains unaffected by this noise and will learn to go to the three modes, one of which is our target location. This would not be the case for methods that use this goal selector as a reward function to run model-free RL, due to its convergence to local maxima without reaching the target goal.

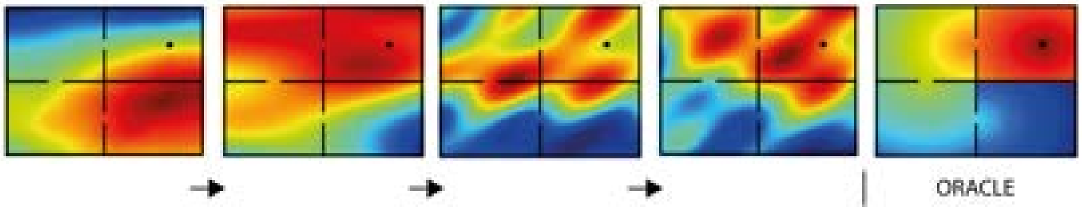


Figure 3.7: Evolution of the learned goal selector when the distance for the synthetic human has a noise of 1.

FewHug can learn when no labels are provided. This property of FewHug is because of the self-supervised learning used to train the policy but also a result of using a parametric goal selector as compared to directly selecting goals of interest as done in [21], which will not have this advantage. From Figure D.5 we observe that a parametric goal selector has the capacity to generalize while, by definition, a non-parametric goal selection [21] will not. Thereafter, using a parametric reward model that has non-degenerate extrapolation can lead to significantly more frontier expansion. In Figure 3.9 we show how our method succeeds in reaching the final goal room even if the goal selector has stopped training when the agent enters the previous to the last room. Freezing the goal selector at different points in the learning of the policy and how long it takes to learn a successful policy. We see that an earlier stop in the training leads to an increase in the timesteps needed to succeed. However, even if we stop in the second room, our method is still very good at quickly finding a successful policy, which shows how robust it is against incomplete goal selectors. This would not be the case for methods that run RL on the learned reward functions (as DDL, and RL from Human Preferences). The policy still succeeds thanks to the added random exploration, the self-supervised nature of GCSL, and a small probability of sampling the final goal.

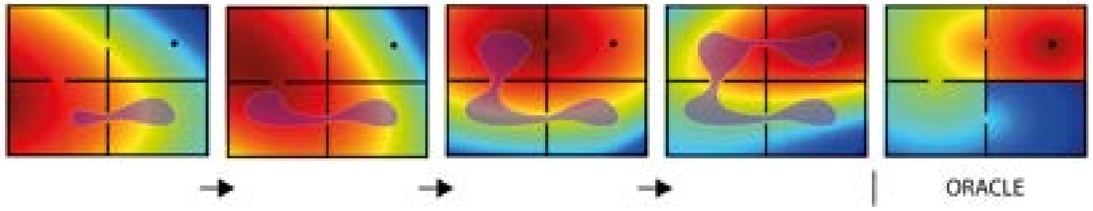


Figure 3.8: Progress of goal selector learning in the four rooms environment as learning progresses it gets closer to the target (oracle on the right). The purple area represents the visited states by the agent at that point. We observe that the goal selector provides extrapolation which will help the training with fewer annotations.

Querying the annotators for labels less frequently makes FewHug more efficient in terms of label efficiency. We observe in Figure 3.6 how with lower frequencies we get fewer labels and we can leverage the generalization of the parametric goal selector to achieve the task with fewer queries. Nevertheless, we note that this reduction in supervision comes at a cost of more episodes needed to train the policy, see Figure D.3.

FewHug is robust to misleading labels. One of the main benefits of using hindsight relabeling to train the policy is that this self-supervised way of learning will allow the policy to continue learning outside of the high-reward regions of the goal selector. A direct implication is that the policy can still learn to reach the goal even if the goal selector has been trained adversarially. Trained adversarially means that wrong labels have been given, trying to lead the agent into the wrong places. To prove this property, we train a goal selector with labels that misguide it to the room before the final one. Even with this misleading goal selector, we observe in Figure 3.9, the agent still manages to get to the goal room when commanded to.

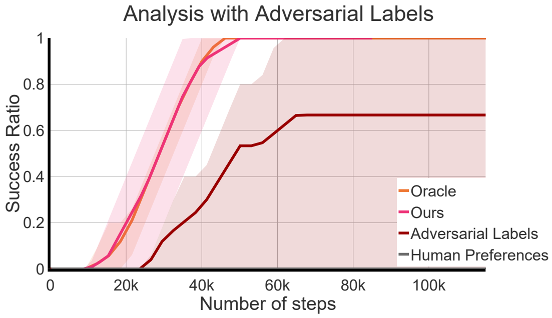


Figure 3.9: Analysis of FewHug. Left: we show it is robust to receiving adversarial labels, where it can still get to the goal.

Metric	Percentage
Current country of Residence	
USA	41.3% (45)
Spain	30.3% (33)
India	8.3% (9)
Germany	6.4% (7)
Canada	2.8% (3)
France	2.8% (3)
Singapore	1.8% (2)
China	1.8% (2)
Andorra	0.9% (1)
Austria	0.9% (1)
Ireland	0.9% (1)
Switzerland	0.9% (1)
United Kingdom	0.9% (1)
Prefer not to say	0% (0)
Gender	
Male	58.7% (64)
Female	39.4% (43)
Non-binary	1.8% (2)
Prefer not to answer	0% (0)
Age group	
18-24	48.6% (53)
25-34	24.8% (27)
35-44	7.3% (8)
45-54	11.0% (12)
55-64	7.3% (8)
65+	0.9% (1)
Prefer not to answer	0% (0)
Education	
Graduate or professional degree	39.4% (43)
College degree	33.9% (37)
High school or some college	20.2% (22)
Other	12.8% (14))
Prefer not to say	2.8% (3)

Table 3.1: Demographics on the participants of the crowdsourced data collection experiment

4

Conclusion

In this work, we introduced FewHug, a technique for solving goal-reaching problems with a challenging element of exploration by leveraging small amounts of human-in-the-loop feedback. We show how binary human preferences can serve to guide goal sampling for goal-reaching problems, leading to directed frontier expansion. In doing so, our proposed technique is able to not just perform indiscriminate exploration or rely on hindsight generalization. Instead, it is able to rely on binary human preferences to train a parametric model of state-goal distances, which can serve to guide frontier expansion towards the goal. We show that this simple technique is able to solve difficult exploration problems for various control tasks in both the real world, learning on real hardware, and in simulation. We additionally ran crowdsourcing experiments with more than 100 human annotators living all over the world to solve the kitchen manipulation benchmark. Finally, we provide ablations and analysis on the design decisions made in FewHug.

There are a number of directions for future work that are very exciting building on FewHug. Firstly, extending to run harder tasks on real robots and in reset-free environments would be very exciting.

5 Contribution

The work presented in this Master Thesis will be submitted as a conference paper to the Conference on Neural Information Processing 2023. For which the author list is the following: Marcel Torné Villasevil, Max Balsells i Pamies, Zihan Wang, Tao Chen, Pulkit Agrawal, Abhishek Gupta. I am the first author of this work, I worked on the algorithm, the experiments in simulation, the experiments from human annotators, I developed the interface system and worked on the writing of the paper. Max worked on adapting the algorithm to work from images as state space and on the real robot experiments. Prof. Abhishek Gupta and Prof. Pulkit Agrawal were the main supervisors of the work, contributing with ideas, helpful feedback and the writing. Finally, Zihan and Tao provided insightful comments on the writing of the manuscript. Tao also supervised me along the project providing helpful advice.

A

Benchmarks

In this section, we give more details on the benchmarks used to compare our method with the baselines. All of these benchmarks are variations of benchmarks presented in previous work. In general, we have made them harder to showcase the benefits of our method. More concretely, for each method, we will give an overview of the difficulties it has and we will present the reward function we designed to provide synthetic labels in our experiments.

1. **Four rooms (small 2D Navigation)**: We consider goal-reaching problems in a 2-D navi-

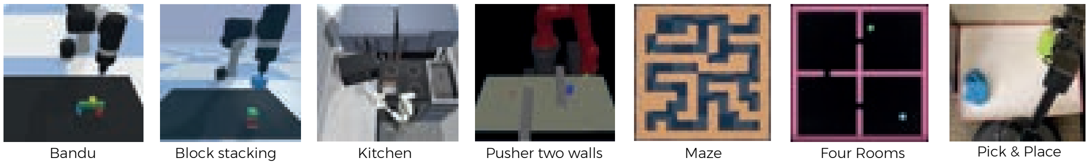


Figure A.1: Benchmarks used to compare FewHug against the baselines.

gation problem in a four rooms environment, shown in Fig A.1. The challenge in this environment is navigation through obstacles, which are unknown without exploration. The agent is initialized in the bottom right room and the goal is sampled from the top right room. The state observation of this environment is the absolute position of the agent in the world, i.e. a vector (x, y), and the action space is discrete with 9 possible actions, encoding 8 directions of movement (parallel to the axis and diagonally), plus a non-move action. To solve this benchmark the agent needs to traverse the two intermediate rooms to get to the target room, traversing a total of four rooms. The reward function in this case is the shaped distance between the state and the goal. This benchmark is a modification of the benchmarks proposed by [18].

2. **Maze (large 2D Maze Navigation)**: We consider a second 2-D navigation problem in a maze environment. The additional challenge in this environment compared to the previous one relies upon having a longer horizon (see Figure B.1). The agent is initialized in the green dot and has to reach the red dot. The state space is the absolute position of the agent in the maze, i.e. a vector (x, y), and the action space is the same as in the Four rooms one, i.e. discrete with dimension 9. The reward function in this case is the shaped distance between the state and the goal.

3. **Pusher two walls**: This is a robotic manipulation problem, where a Sawyer robotic arm pushes an obstacle in an environment with multiple obstacles. The puck and arm start in the

configuration seen in Fig A.1. The task is considered successful if the robotic manipulator brings the puck to the goal area, marked with a red dot. The state space of this environment consists of the position of the puck and the position of the arm. The action space is the control of the position of the robotic arm. It is also a 9-dimensional discrete action space where each one corresponds to a delta change in the position in 2D. This benchmark is a modification of the benchmarks proposed by[18]. The reward function designed for this environment is the following:

$$r = max(distance\_puck\_finger, 0.05) + distance\_puck\_goal$$

4. **Sequential Kitchen Manipulation**: This benchmark is a harder robotic manipulation task where apart from being long horizon the agent needs to show three different skills to solve the task. We operate a 7 DoF Franka robot arm in a simulated kitchen, manipulating different cabinets, sliding doors, and other elements of the scene. The observation space consists of the position of the end effector and its rotation together with the joint states of the target objects. The action space consists in controlling the end effector position in 3D, we discretize it so the dimension is 27, and the control of the gripper and rotation of the arm. In our evaluation, we consider tasks where the goal is to sequentially manipulate three elements in the kitchen environment - the sliding cabinet, the microwave and the hinge cabinet to target configurations. The reward function we use is the following:

$$r = \begin{cases} -\text{distance(arm, hinge cabinet)} - |\text{hinge cabinet target joint - hinge cabinet current joint}|, & \text{if slide cabinet and microwave opened} \\ -\text{distance(arm, microwave hinge)} - |\text{microwave target joint - microwave current joint}| - \text{bonus}, & \text{if slide cabinet opened} \\ -\text{distance(arm, slide cabinet hinge)} - |\text{slide cabinet target joint - slide cabinet current joint}| - 2\text{bonus}, & \text{otherwise} \end{cases} \tag{A.1}$$

5. **Block Stacking**: This domain is another long horizon robotic manipulation task, we operate a 6 DoF UR5 robot arm with a suction gripper as an end effector in a simulated tabletop configuration, stacking blocks. The observation space consists of the position of the end effector and the position of each block in 2D, and a bit indicating whether the hand is holding a block. This is a continuous action space domain with dimension 2, where the agent will predict a grasp position if it does not hold an object and a drop position if it is holding an object. We consider the goal to be accomplished if the three blocs are stacked in the correct order (red, green, blue) on the correct fixed place on the table. The reward function is the following:

$$r = \begin{cases} -\text{distance(arm, blue block) - distance(blue block, target goal)}, \text{if red and green block at position} \\ -\text{distance(arm, green block) - distance(green block, target goal)} - \text{bonus}, \text{if red block at position} \\ -\text{distance(arm, red block) - distance(red block, target goal)} - \text{2bonus}, \text{otherwise} \end{cases} \tag{A.2}$$

6. **Bandu**: This domain is very similar to the block stacking. We operate a 6 DoF UR5 robot arm with a suction gripper as an end effector in a simulated tabletop configuration. The observation space consists of the position of the end effector and the position of each block in 2D, and a bit indicating whether the hand is holding a block. This is a continuous action space domain with dimension 2, where the agent will predict a grasp position if it does not hold an object and a drop position if it is holding an object. We consider the goal to be accomplished if the four blocs are stacked in the target configuration building the castle like

structure seen in Figure A.1. The reward function is the following:

$$r = \begin{cases} -\text{distance(arm, yellow star) - distance(yellow star, target yellow star)}, & \text{if all except star at position} \\ -\text{distance(arm, green block) - distance(blue green block, target green block)} - \text{bonus}, & \text{if red and blue blocks at position} \\ -\text{distance(arm, blue triangle) - distance(blue triangle, target blue triangle)} - 2\text{bonus}, & \text{if red cylinder at position} \\ -\text{distance(arm, red cylinder) - distance(red cylinder, target red cylinder)} - 3 - \text{bonus}, & \text{otherwise} \end{cases} \tag{A.3}$$

More details about how these benchmarks were run, such as the number of episodes we ran the benchmarks for, are presented in Section B.0.2

B

Implementation Details

B.0.1 Networks with Fourier Features

Seeing the complexity of our benchmarks, where we can have non-smooth reward landscapes for the goal selector. For example, in the four rooms environment, between one side and the other of the right rooms, the reward changes significantly and abruptly. Adding Fourier Features has been shown to work well for fitting these landscapes[39]. For this reason, we used them in some of our experiments, as detailed in Section B.0.2. More precisely, when used, we added an additional layer with Fourier features of size 40 times the input dimension.

B.0.2 Training details

The details of the parameters with which the results have been obtained will be disclosed in this section. In particular, Table B.1 depicts the parameters used for the different benchmarks, while Table B.2 contains the hyperparameter configuration used for the different algorithms.

	Four rooms	Maze	Pushing around Obstacles	Kitchen Manipulation	Block Stacking
Steps per trajectory	50	250	100	100	8
Label from last k steps	10	50	10	20	8

Table B.1: Benchmark-related parameters

Parameter	Value
Shared (to those that apply)	
Optimize	Adam
Discount factor (γ)	0.99
Reward model architecture	MLP(256, 256)
Use Fourier in the reward model	True
Buffer size reward model	1000
Steps per reward model update	1000
GCSL, Oracle and Ours	
Learning rate	$5 \cdot 10^{-4}$
Batch size	100
Policy architecture	MLP (400, 600, 600, 300)
Steps per policy update	5000
Use Fourier in the policy model	True
Buffer size rollout	1000
Max gradient norm	5
Last trajectories to be labeled	1000
Human preferences Same parameters as[36] plus/except	
Learning rate	$5 \cdot 10^{-4}$
Batch size	100
Policy architecture	MLP (256, 64)
Steps per policy update	5000
Use Fourier in the policy model	False
Buffer size rollout	1000
Max gradient norm	5
Last trajectories to be labeled	1000
DDL	
Learning rate	$5 \cdot 10^{-4}$
Batch size	256
Buffer Size	$2 \cdot 10^{4}$
Policy architecture	MLP (256, 256)
Steps per update	1000
PPO Same parameters as[36] plus	
Buffer size	8192
Policy architecture	MLP (400, 600, 600, 300)

Table B.2: Hyperparameters setting for the algorithms

C

Further analysis of the baselines

C.0.1 Failure modes of exploration algorithms for goal-reaching

For the sake of concreteness, we will study two simple schemes from prior work on solving goal-reaching problems —self-supervision via goal conditioned supervised learning [18] (as described in Section 2.1) and reinforcement learning with density based exploration [29]. Exploration in GCSL relies on generalization of the policy across goals, while density based exploration rewards exploring

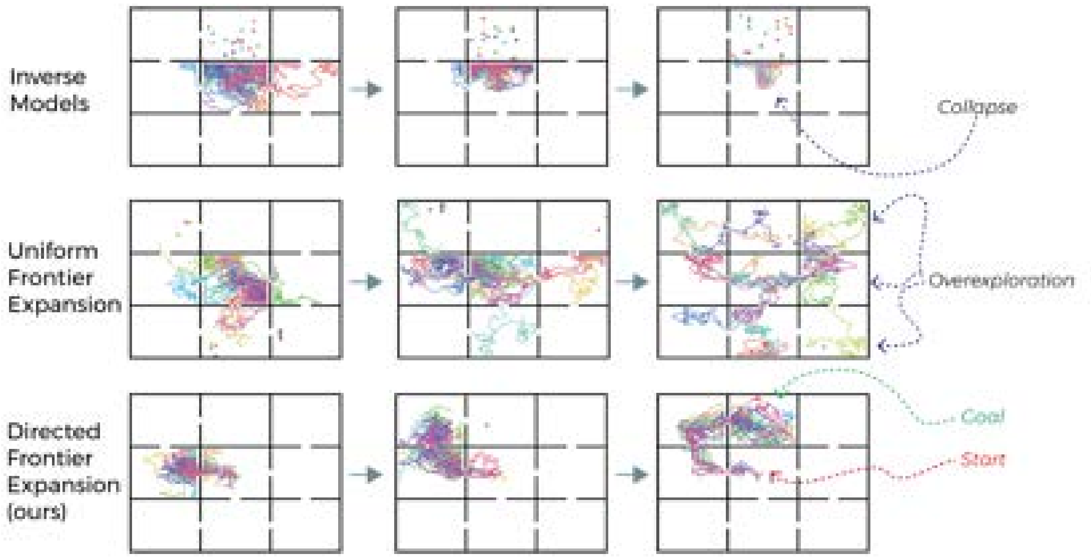


Figure C.1: Failure modes of exploration algorithms for goal-reaching. Inverse models (top) collapses and does not discover the target room (second room at the top). Uniform frontier expansion (middle) does reach the target room, but to get there it visits all possible rooms, since exploration is indiscriminate. Directed frontier expansion (bottom, ours) reaches the target room much faster by leveraging human signal on direction. Training epochs increase from left to right. Each subfigure is an aerial view of a floor with 9 rooms, with multiple trajectories, each one in a different color.

the most novel states. We show these algorithms can fail in different ways for a simple maze environment shown in Fig C.1, where the agent starts in the middle room and must reach goals commanded in the top middle room.

As shown in Fig C.1, GCSL exploration quickly collapses in the maze environment. This can be understood by noticing that self-supervised training on goals in the bottom right corner room or even the bottom left corner room does not extrapolate to the top right corner, where the commanded goals are. Instead of navigating the agent around the walls, the policy generalization suggests that the agent simply go into the wall as shown in Fig C.1.

Exploration methods are meant to tackle this kind of degenerate exploration, by encouraging visitation of less frequently visited goals at the "frontier" of visited states. When applied to the goal-reaching problem, in Fig C.1, we see that while the exploration is not degenerate, exploration is indiscriminate in that it explores both sides of the maze even though commanded goals are only down one particular path. While this will eventually succeed, it incurs a significant cost of redundant exploration by going down *redundant* paths.

This suggests that frontier expansion is needed like exploration methods, but should ideally be done in a directed way towards goals of interest. In Figure C.1 we see how this directed exploration could be useful and reduce sample complexity, by removing the need for indiscriminate frontier expansion. We show how a small amount of relatively cheap human feedback can be leveraged to guide this exploration.

C.0.2 Detailed training curves

For some of the runs the plot of the success could be misleading, in the sense that, despite not achieving the goal, the algorithms may still learn how to almost solve the task, or at least gained some knowledge about how to approach it. Figure C.2 shows for each of the runs, the distance to the goal, which corresponds to $-r$ where r is the reward of the corresponding benchmark, as described in Section A.

For example, by looking at Figure C.2, we can see that despite the fact that the Human Preferences wasn't able to complete some of the tasks, such as Four Rooms, Pusher with walls or Maze, it still got some insight on how to approach it, getting much closer to the goal than the other methods that failed.

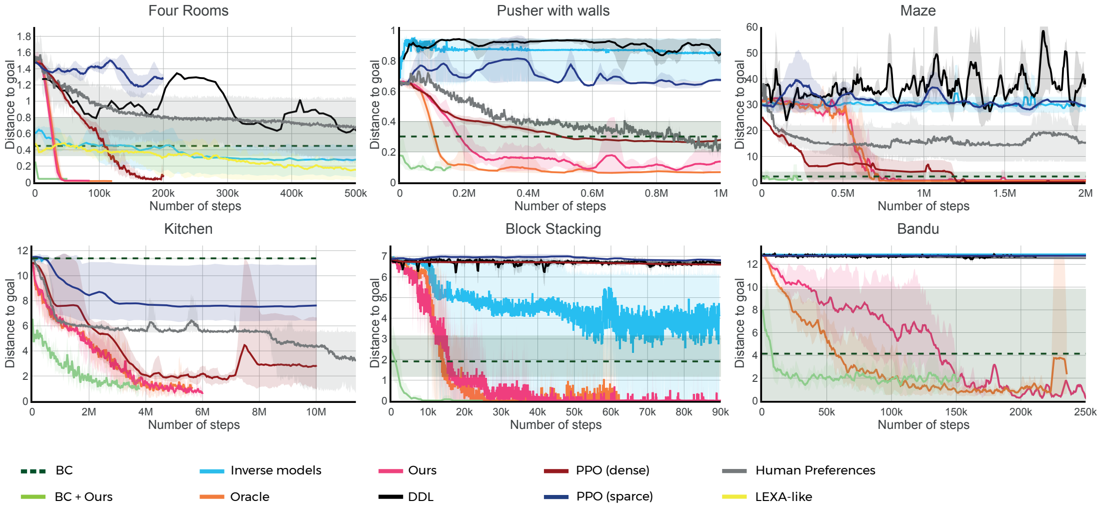


Figure C.2: Distance to the goal for each method on different benchmarks. We note that the LEXA-like exploration strategy was only implemented on the four rooms benchmark.

Benchmark	Oracle	Ours	GCSL	Human Preferences	DDL	PPO (sparse)	PPO (dense)	LEXA style
4 rooms	0.02 ± 0.01	0.02 ± 0.00	1.15 ± 0.67	0.48 ± 0.39	0.45 ± 0.28	1.45 ± 0.13	0.05 ± 0.02	0.13 ± 0.18
Maze	0.4 ± 0.3	0.8 ± 0.3	29.6±2.2	18.5 ± 5.6	37.2±5.3	30.4±0.7	0.0 ± 0.2	-
Pusher	0.06 ± 0.00	0.11 ± 0.04	0.85 ± 0.11	0.26 ± 0.03	0.69 ± 0.06	0.72 ± 0.06	0.27 ± 0.00	-
Kitchen	2.0 ± 0.6	4.5 ± 1.1	34.9±0.3	5.5 ± 0.6	16.4±4.9	32.5±2.3	21.9±3.9	-
Stacking	0.1 ± 0.2	0.0 ± 0.0	4.1 ± 2.3	6.5 ± 0.1	6.6 ± 0.1	6.7 ± 0.0	6.6 ± 0.0	-

Figure C.3: Average distance and standard deviation across 4 seeds for the different baselines we implemented to compare against HugRL. We see that HugRL consistently succeeds (in bold) to solve all benchmarks when most other baselines do not. The oracle would be the upper bound that we could hope to achieve, since in this case labels are provided all the time, and the goal selector is substituted by a precise distance function..

D

Further Analysis and Ablations

D.1 Compatibility with Learning from Demonstrations

FewHug is compatible with Learning from Demonstrations. To pretrain the policy we can initialize the replay buffer with the expert trajectories and run several iterations of policy learning with hindsight relabelling on the expert trajectories. We can also pretrain the goal selector by taking random states from a single trajectory and labelling the state achieved later in time as being closer to the goal.

We also initialize the goal selector buffer with such tuples of achieved states and label and pretrain the goal selector on this. In Figure D.1, we show the effect that adding more demonstrations has on the overall training of the policy. We see that more demonstrations lead to a better start of the policy at the 0th timestep. We also observe that using FewHug we can fine tune and improve considerably the performance of the policy.

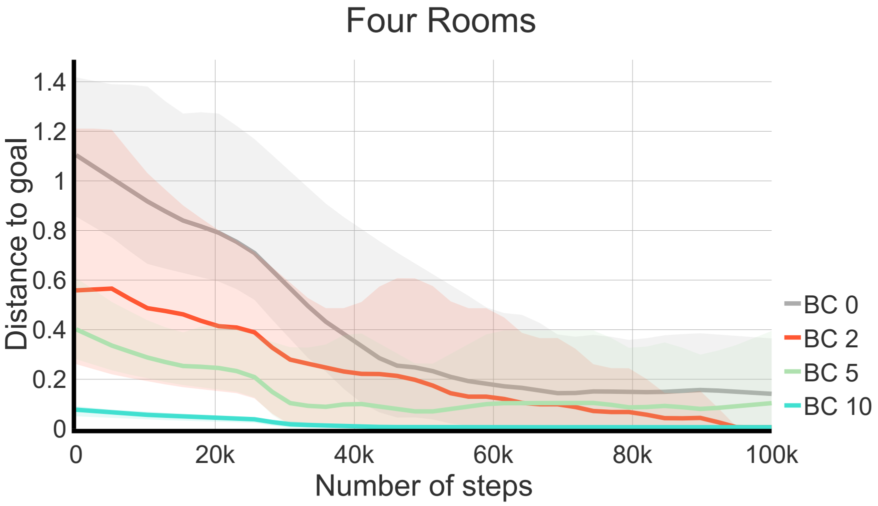


Figure D.1: The figure depicts the distance to the goal in the Four Rooms environment when using a policy pre-trained via Behaviour Cloning with 0, 2, 5, and 10 demonstrations, respectively. We see that using BC on a small number of demonstrations helps to boost the performance of our method. Also, notice that BC wouldn't achieve success (distance < 0.05) in any of the cases due to compounding errors which leads to covariant shift. However, HugRL solves these compounding errors within a small number of steps.

D.2 Effect of frequency and amount of querying on the training

We aim to understand how sensitive the reward model is to the frequency of human supervision. In the four rooms domain, we ablate the frequency at which we query the human for feedback. The results are depicted in Figure D.3. We observe a clear tradeoff between needing fewer labels to succeed against needing more timesteps. Meaning that if we query more frequently, we will need fewer timesteps to succeed and vice versa.

On the other side, we also try to understand how much feedback is needed every time we query the human. In Figure D.2, we keep the querying frequency fixed and we vary the amount of feedback. We observe that we can go as low as 5 queries per batch, and the performance is similar to 20 and 100 labels. Showing that too many queries bring duplicated information to the goal selector training and hence those annotations become useless. However, if we query too little, for example providing 1 label, then this is not enough, degrading the performance significantly.

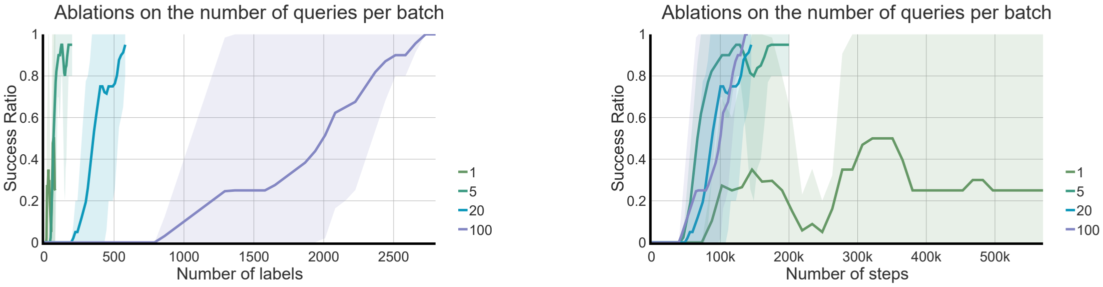


Figure D.2: On the right we show the number of steps needed to succeed in the four rooms benchmark depending on the number of comparisons queried per batch. On the left, we show the number of labels needed to succeed, again depending on the amount of queries. These experiments are done in the Four Rooms benchmark

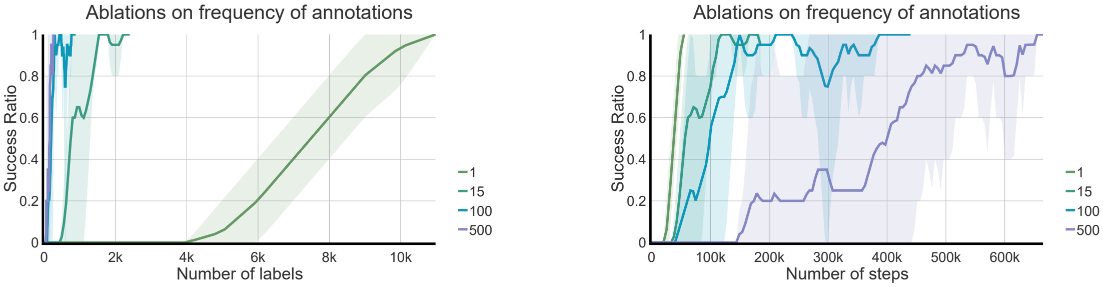


Figure D.3: On the left/right we show the number of labels/timesteps needed to succeed when varying the query frequency. 1, 15, 100, and 500 are the number of episodes between each period of querying the human for annotations.

D.3 Effect of quality of the underlying reward function

In this section, we analyze the performance of our method in the Four Rooms environment when dealing with a simplified version of feedback. In particular, we only return feedback if the given queried states have a distance difference of at least d with respect to the goal. For context, in this environment, 0.5 is approximately the distance between the center of two consecutive rooms, so using $d \geq 0.5$ is roughly similar to using the room number as a reward function. Therefore, in this experiment, we can see that, even with very simple reward functions, we can still get some insight on how to solve the task, though at the expense of clearly slower convergence. In particular, we can see how coarser reward functions lead to worse performances. This also helps us understand what happens in scenarios in which it is hard for humans to compare states that are similarly good for the purpose of achieving the required goal.

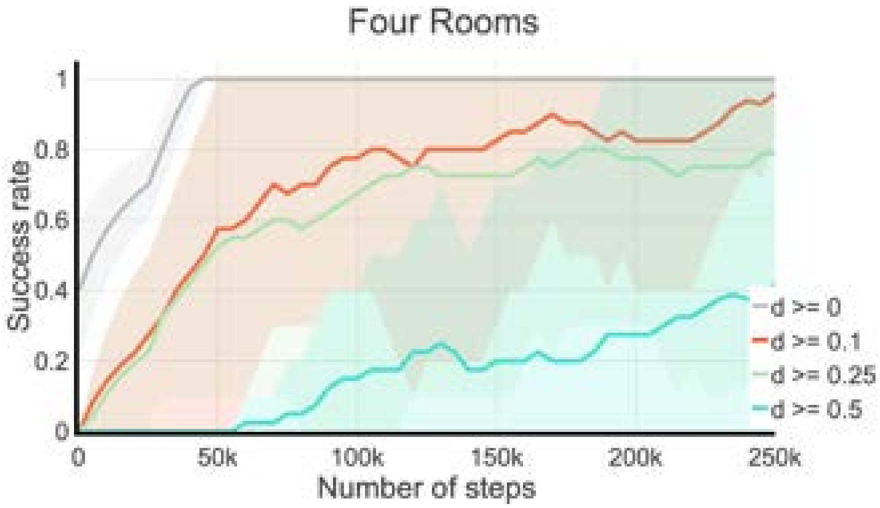


Figure D.4: Effect of a simplified feedback given, where two states will only be labeled if their distance is greater than d, otherwise the synthetic human will return a ”don’t know” as a label, which is ignored by our algorithm.

D.4 Analysis of the Goal Selector learning

Firstly we present a qualitative analysis, where we show visualizations of the learned goal selector as different rooms are discovered during the learning process in the four-rooms domain in Fig D.5. The goal selector model shows nontrivial extrapolation and can potentially provide guidance even beyond the set of states it is trained on.

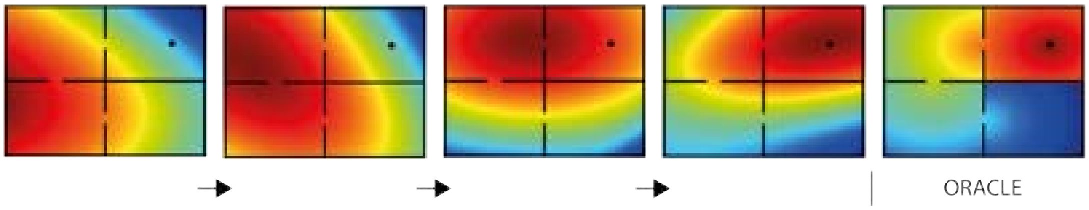


Figure D.5: The goal selector learns and converges to a result close to the oracle (rightmost) as epochs increase (left to right). We observe how this goal selector gets updated iteratively as the frontier expands. Colder colors mean a lower reward for that state, whereas warmer colors mean a higher reward for that state, in this case, this is equivalent to the distance to the goal.

We perform a second quantitative analysis, presented in Figure D.6 where we explore how accurate the goal selector is, depending on the number of queries it has been trained with. In particular, we tested it in the Four Rooms environment by training the goal selector using pairs of states sampled uniformly. During evaluation, given two states which are less than d units apart, we compute the accuracy for which the model is able to pick the closest state to the goal. This allows us to see that the model is able to, given two states, determine which one is the closest to the goal, even when the given states are very close together and even when trained with just a handful of queries. For context, bear in mind that the distance from the initial state to the goal is 1.6 units.

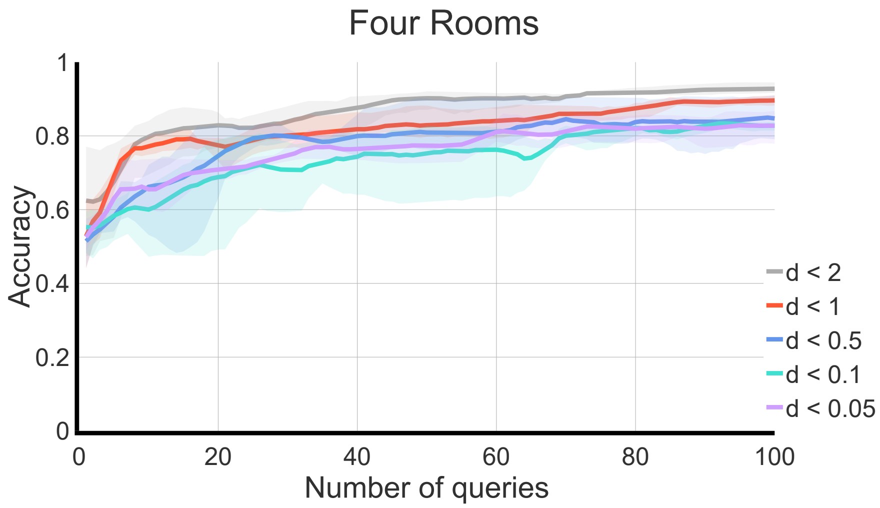


Figure D.6: Analysis of the accuracy of the goal selector depending on the distance d between the states given a certain number of queries sampled uniformly from the state space.

D.5 Effect of terminal exploration in frontier expansion

How important is terminal exploration to frontier expansion? To understand the importance of random exploration after the breadcrumb goal is reached (as described in Section Section 2.2.1), we evaluate the learning progress with and without this terminal exploration in Fig D.7. We see that if no exploration is added our method fails, the reason being that new states are only visited with the stochasticity of the policy which will decrease over time as it overfits and hence the visitation of new states will also decrease. For this reason, without final random exploration, the policy will get stuck without expanding its frontier anymore.

How important is taking out redundant steps? One of the tricks to make this method work is to take out redundant steps. We define redundant steps as those that produce no change in the observation space, one example would be advancing towards a wall when already in contact with it.

In Figure D.8 we see the resulting performance with and without taking out redundant steps. In

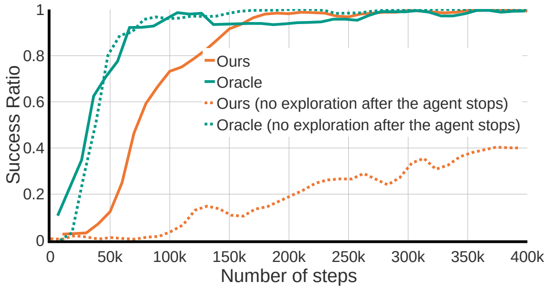


Figure D.7: Effect of the use of exploration after reaching the commanded goal on the performance

particular, we can see that, even though our method can still reach the goal when having redundant steps, it converges much slower than when we remove them, highlighting the importance of taking them out.

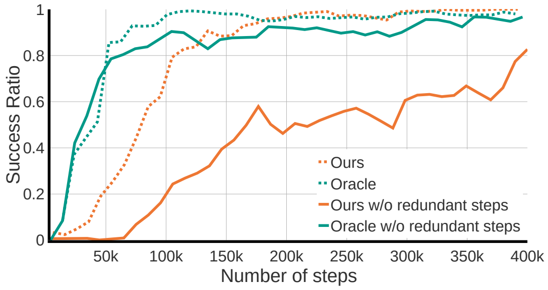


Figure D.8: Effect of the use of exploration after reaching the commanded goal on the performance

E

Details on the Human Experiments

E.1 Human experiment on pusher and 2D navigation

In this section, we give more details on how we ran the human experiment on the pusher and 2D navigation tasks. We designed a simplified interface shown in Figure E.1. We can see the two states to be compared in blue and red and the goal we aim to achieve in green. Then the annotator has to decide which one of the two states is closer to the given goal and provide feedback by clicking either on the blue or red button. In the case in which the annotator is undecided, they can click on the gray button that simply skips the current case. Finally, if the annotator does not provide any feedback after 30 seconds of being presented with the scenario, we skip the current batch of labeling and continue with training the policy. With this, we can take advantage of the properties of our method and continue training the policy even when no labels are given.

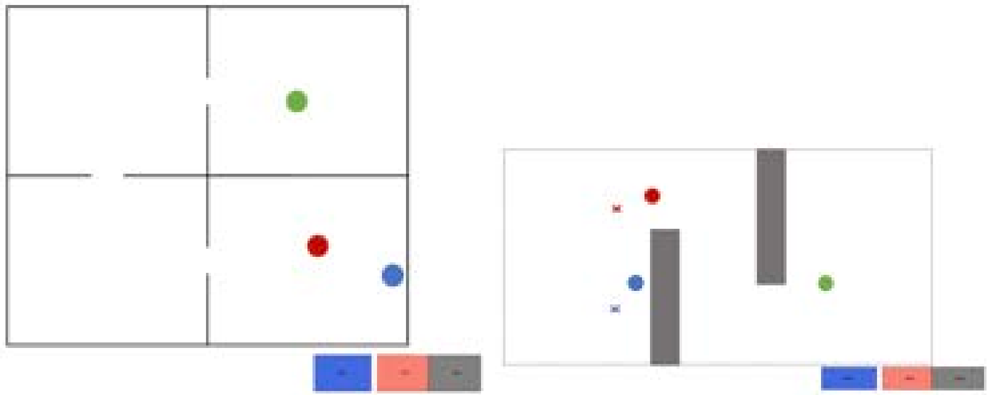


Figure E.1: Interfaces for maze navigation (left) and pusher walls (right)

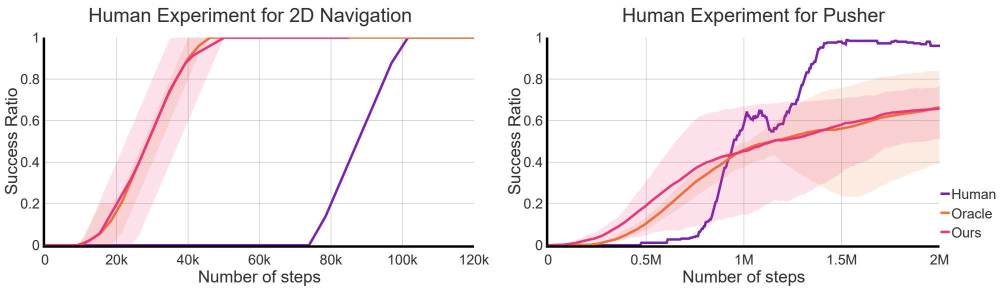
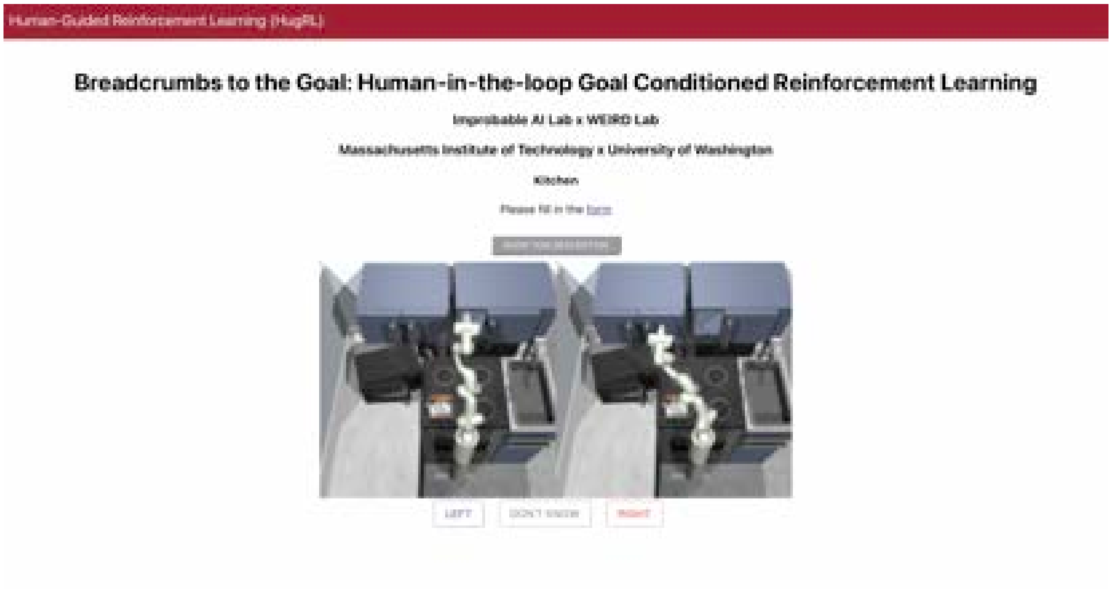


Figure E.2: Learning Progress with Human-in-the-Loop Feedback for maze navigation (left) and pusher walls (right)

In Figure E.2 we share again the results obtained with the human experiment on a larger scale. We ran both experiments using the same frequency of labeling and number of labels per batch. In particular, we labeled every 50 rollout trajectories and queried the annotators for 20 labels. These parameters were identified through empirical experiments.

E.2 Crowdsourced Human Experiment on the Kitchen

In this section, we will go more into the details of the human experiments run on the kitchen benchmark. First, in this experiment, we were not able to provide a simplified representation of the state space as we did for the pusher and 2D navigation tasks. Hence, we directly provided the state images obtained during the rollouts of our policy in the environment, as seen in Figure E.3. The interface remained very similar to the previous experiments, we continued having the left/right buttons for selecting which state is preferred by the annotator and the "don't know" button in case they do not see it clearly.



Human-Guided Reinforcement Learning (HugRL)

Breadcrumbs to the Goal: Human-in-the-loop Goal Conditioned Reinforcement Learning

Improbable AI Lab x WEIRD Lab

Massachusetts Institute of Technology x University of Washington

Kitchen

Please fill in the form

Figure E.3: Interface for the crowdsourcing experiment. We present two images to the user and ask for their feedback on which one is closer to the target goal. The target goal is displayed when clicking on "Show Task Description" together with additional information about how to user the interface and was is expected from the user.

The system behind this interface is more complex since we needed to make it support concurrent users from all over the world, labeling from their own places. Overall the structure of the system consisted in a frontend website that was running on one of our servers built with NodeJS. The interface was handling the responses from the user and sending them to the backend, as well as requesting the backend for additional states to compare. Then, we had the backend, also running on our servers, and was built using FastAPI. The backend was in charge of running the algorithm in parallel to accepting the answers coming from the interface through a public API, as well as sending new pairs of images to label when requested.

Metric	Percentage
Nationality	
Spain	26.6% (29)
USA	20.2% (22)
India	9.2% (10)
Germany	8.3% (9)
China	7.3% (8)
France	4.6% (5)
Mexico	3.7% (4)
Colombia	2.8% (3)
Switzerland	1.8% (2)
Hong Kong	1.8% (2)
Canada	1.8% (2)
Uruguay	0.9% (1)
Singapore	0.9% (1)
Russia	0.9% (1)
Ireland	0.9% (1)
Lebanon	0.9% (1)
South Korea	0.9% (1)
Sweden	0.9% (1)
Andorra	0.9% (1)
Puerto rico	0.9% (1)
Israel	0.9% (1)
Prefer not to say	0.9% (1)
Ethnicity	
Hispanic, Latino or Spanish	38.5% (42)
Asian	28.4% (31)
White or Caucasian	24.5% (27)
Middle Eastern or North African	3.7% (4)
South-east Asian	2.8% (3)
Black or African American	0.9% (1)
Perfer not to say	0.9% (1)

Table E.1: Demographics on the participants of the crowdsourced data collection experiment

References

[1] Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., & Zaremba, W. (2017a). Hindsight experience replay. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA* (pp. 5048–5058).

[2] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., & Zaremba, W. (2017b). Hindsight experience replay. In *Advances in Neural Information Processing Systems* (pp. 5048–5058).

[3] Azar, M. G., Osband, I., & Munos, R. (2017). Minimax regret bounds for reinforcement learning. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research* (pp. 263–272).: PMLR.

[4] Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems* (pp. 1471–1479).

[5] Biyik, E. (2022). Learning preferences for interactive autonomy. *CoRR*, abs/2210.10899.

[6] Biyik, E. & Sadigh, D. (2018). Batch active preference-based learning of reward functions. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research* (pp. 519–528).: PMLR.

[7] Brafman, R. I. & Tennenholtz, M. (2002). R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3, 213–231.

[8] Burda, Y., Edwards, H., Storkey, A., & Klimov, O. (2018). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

[9] Cabi, S., Colmenarejo, S. G., Novikov, A., Konyushkova, K., Reed, S. E., Jeong, R., Zolna, K., Aytar, Y., Budden, D., Vecerík, M., Sushkov, O., Barker, D., Scholz, J., Denil, M., de Freitas, N., & Wang, Z. (2020). Scaling data-driven robotics with reward sketching and batch reinforcement learning. In M. Toussaint, A. Bicchi, & T. Hermans (Eds.), *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*.

[10] Cederborg, T., Grover, I., Jr., C. L. I., & Thomaz, A. L. (2015). Policy shaping with human teachers. In Q. Yang & M. J. Wooldridge (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015* (pp. 3366–3372).: AAAI Press.

[11] Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., & Amodei, D. (2017a). Deep reinforcement learning from human preferences.

[12] Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., & Amodei, D. (2017b). Deep reinforcement learning from human preferences. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *NeurIPS*.

[13] Coumans, E. & Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning. In *GitHub Repository* (pp. 5026–5033).

[14] Davchev, T., Sushkov, O. O., Regli, J., Schaal, S., Aytar, Y., Wulfmeier, M., & Scholz, J. (2022). Wish you were here: Hindsight goal selection for long-horizon dexterous manipulation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*: OpenReview.net.

[15] Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., & Clune, J. (2019). Go-explore: a new approach for hard-exploration problems. *CoRR*, abs/1901.10995.

[16] Eysenbach, B., Salakhutdinov, R., & Levine, S. (2019). Search on the replay buffer: Bridging planning and reinforcement learning. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (pp. 15220–15231).

[17] Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477.

[18] Ghosh, D., Gupta, A., Fu, J., Reddy, A., Devin, C., Eysenbach, B., & Levine, S. (2019). Learning to reach goals without reinforcement learning. *CoRR*, abs/1912.06088.

[19] Gupta, A., Kumar, V., Lynch, C., Levine, S., & Hausman, K. (2019). Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In L. P. Kaelbling, D. Kragic, & K. Sugiura (Eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka,*

Japan, October 30 - November 1, 2019, Proceedings, volume 100 of *Proceedings of Machine Learning Research* (pp. 1025–1037).: PMLR.

[20] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290.

[21] Hartikainen, K., Geng, X., Haarnoja, T., & Levine, S. (2019). Dynamical distance learning for unsupervised and semi-supervised skill discovery. *CoRR*, abs/1907.08225.

[Kaelbling] Kaelbling, L. P. : Citeseer.

[23] Knox, W. B. & Stone, P. (2008). TAMER: Training an Agent Manually via Evaluative Reinforcement. In *IEEE 7th International Conference on Development and Learning*.

[24] Lee, K., Smith, L. M., & Abbeel, P. (2021). PEBBLE: feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research* (pp. 6152–6163).: PMLR.

[25] Levy, A., Konidaris, G. D., Jr., R. P., & Saenko, K. (2019). Learning multi-level hierarchies with hindsight. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*: OpenReview.net.

[26] Liu, M., Zhu, M., & Zhang, W. (2022). Goal-conditioned reinforcement learning: Problems and solutions. In L. D. Raedt (Ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022* (pp. 5502–5511).: ijcai.org.

[27] Logan, G. D. (2002). An instance theory of attention and memory. In *Psychological Review, 109(2), 376–400* (pp. 5026–5033).

[28] Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., & Sermanet, P. (2019). Learning latent plans from play. In L. P. Kaelbling, D. Kragic, & K. Sugiura (Eds.), *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research* (pp. 1113–1132).: PMLR.

[29] Mendonca, R., Rybkin, O., Daniilidis, K., Hafner, D., & Pathak, D. (2021). Discovering and achieving goals via world models. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual* (pp. 24379–24391).

[30] Murali*, A., Chen*, T., Alwala*, K. V., Gandhi*, D., Pinto, L., Gupta, S., & Gupta, A. (2019). PyRobot: An open-source robotics framework for research and benchmarking. *CoRR*, abs/1906.08236.

[31] Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J., & Levine, S. (2017). Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017* (pp. 2146–2153).: IEEE.

[32] Nair, A., Pong, V., Dalal, M., Bahl, S., Lin, S., & Levine, S. (2018). Visual reinforcement learning with imagined goals. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (pp. 9209–9220).

[33] Osband, I., Blundell, C., Pritzel, A., & Roy, B. V. (2016). Deep exploration via bootstrapped DQN. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (pp. 4026–4034).

[34] Pong, V., Dalal, M., Lin, S., Nair, A., Bahl, S., & Levine, S. (2020). Skew-fit: State-covering self-supervised reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research* (pp. 7783–7792).: PMLR.

[35] Ross, S., Gordon, G., & Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 627–635).: JMLR Workshop and Conference Proceedings.

[36] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[37] Sharma, P., Sundaralingam, B., Blukis, V., Paxton, C., Hermans, T., Torralba, A., Andreas, J., & Fox, D. (2022). Correcting robot plans with natural language feedback. *CoRR*, abs/2204.05186.

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

[39] Yang, G., Ajay, A., & Agrawal, P. (2022). Overcoming the spectral bias of neural value approximation. In *International Conference on Learning Representations*.

This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, "Science Experiment 02", was created by Ben Schlitter and released under cc by-nc-nd 3.0. A template that can be used to format a PhD thesis with this look and feel has been released under the permissive mit (x11) license, and can be found online at github.com/suchow/Dissertate or from its author, Jordan Suchow, at suchow@post.harvard.edu.