Next, we provide additional details of our work. More concretely:

- Appendix A **Real-Robot experiments**: provides more details on the real-robot experiments and on HUGE working from images.
- Appendix B **Real-Human experiments**: shows all the details of the real human experiments presenting results for both the crowdsourcing experiment as well as experiments on more benchmarks with fewer annotators.
- Appendix C **Simulated Benchmarks**: we provide further details on the simulated benchmarks used in this work.
- Appendix D **Baselines**: we provide details about the baselines together with more detailed learning curves of the baselines on the simulated benchmarks.
- Appendix E **Further Analysis and Ablations**: we provide more insights on where the benefits of HUGE come from, as well as provide some ablations on the method.
- Appendix F **Implementation Details**: we provide further implementation details, hyperparameters and resources used.

The code is available at `github.com/...`

A Real-Robot Experiments

HUGE's qualities make it suitable to learn policies directly in the real world. However, we adapted the method with respect to the simulated experiments in Fig 5. The main change consisted in changing the state space to image space instead of point space. Next, we show HUGE works from image space in two of the simulated environments, four rooms and block stacking.

A.1 HUGE from images

Adapting HUGE to work from image space was a trivial process, the goal selector and policy networks were modified introducing a first encoding network consisting of 3 convolutional layers (with stride 2 and kernel size 5) to map the input image to a lower dimension space and then we passed it through an MLP to predict the score and action respectively.

Learning stopping criteria: When using the point state space, we could easily detect whether the policy stopped, indicating it reached the target goal, or that it got stuck, to then start random exploration from there. This could be done by computing the Euclidean distance and setting a small enough threshold. This is more difficult when working from image state space. What we did was train an image classifier $\phi(s_1, s_2)$ that predicts whether the two images correspond to states close in space (i.e. the state corresponding to image s_2 can be reached from the state corresponding to image s_1 within t_{close} timesteps). We trained ϕ by using contrastive learning [11]. In particular, we sampled images from our replay buffer and assigned the corresponding label based on their distance in timesteps: $l(s_i, s_j) = 1$ if $|i - j| \leq t_{\text{close}}$ and 0 if $|i - j| \geq t_{\text{far}}$. Based on the premise that, in most cases, images obtained far away in time, will probably correspond to states that take longer than t_{close} timesteps to reach, if we were to act optimally.

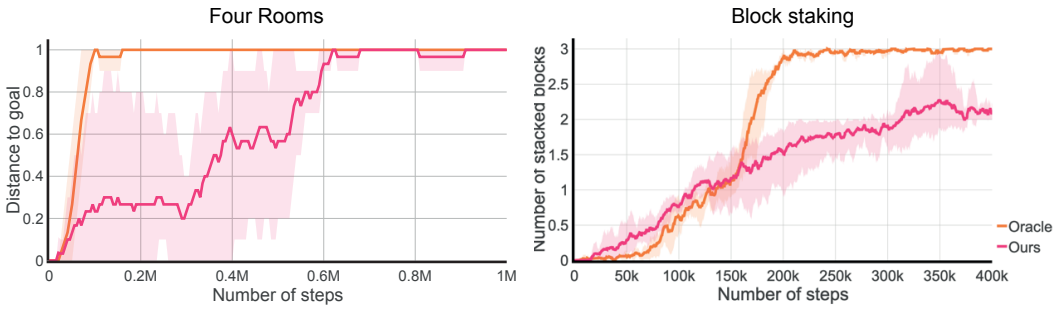

Figure A.1: Success rate for the four rooms (**left**) and block stacking (**right**) using images as input space for both the policy and goal selector.

A.2 Results in the real-world

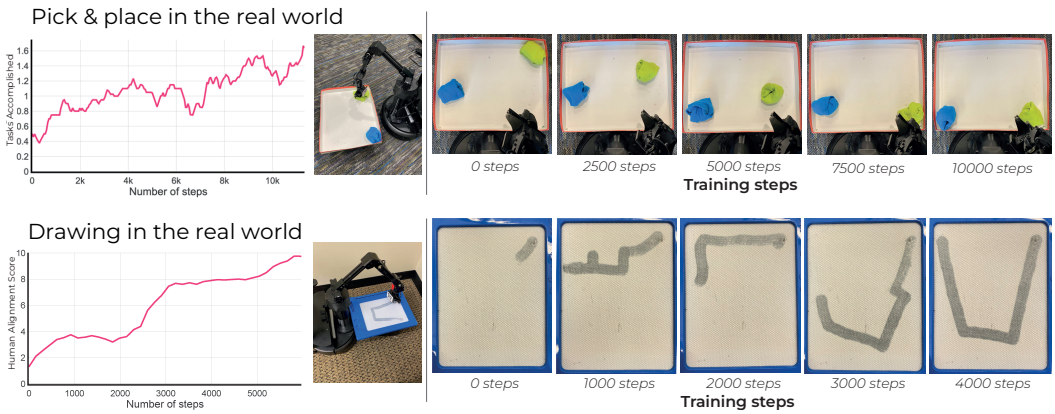

Figure A.2: Accomplished goals at the end of 5 different evaluation episodes along training on the real world to pick and place, and draw the letter U in the real world.

For the real robot experiment, we used a LoCoBot with a WX-200 arm.

Pick and place in the real world: The state space consisted of RGB images of 64×64 pixels, and the action space was continuous with dimension 2, representing an absolute position in the space (x, y) from which to predict a grasping point in even timesteps, or a dropping point in odd timesteps. For the experiment to be succeed in a reasonable amount of time, we pretrained the policy and the goal selector by using 5, sub-optimal demonstrations. The robot was trained for around 30h, during which, 130 labels were provided via the interface shown in B.4 by one annotator. Finally, we used a reset mechanism to pull the socks to the same corners, though, it had some stochasticity.

Drawing in the real world: The state space consisted of RGB images of 64×64 pixels, and the action space was discrete, encoding a total of 5 actions: no movement and moving across the two axis on the plane in polar coordinates (i.e. increasing r, decreasing r and moving a fixed amount clockwise or counterclockwise), to move the end effector with the brush. An episode consisted of 12 timesteps. For the experiment to be ran in a reasonable amount of time, we pretrained the policy and the goal selector by using 5, sub-optimal demonstrations. The robot was trained for around 6h, during which, 150 labels were provided via the interface shown in B.4 by one annotator. Finally, the reset was done by using the erase mechanism in this drawing boards and moving it with the arm by using a script. As a final note, in this environment we had to perform few exploration steps and slowly increase the frontier. This is because in this environment there is only one optimal solution (the actions taken must be exactly the optimal ones, due to the fact that all past actions within the episode will affect the current state of the board), in particular, any non-optimal action will leave a trace, making that trajectory not that useful for the policy to learn from it.

Human Alignment evaluation for drawing in the real world: Designing a reward function for drawing is a hard and tedious labor. HUGE does not need a reward function and we can fully leverage human feedback to learn this behavior as shown in A.2. Without a reward function evaluation cannot be performed either. For this reason, we defined this "Human Alignment Score" that basically consists in querying humans and asking them for a score between 0 and 10 of how well the robot draw the target picture. In the case of the drawing experiments, we asked 2 annotators to label the performance of the robot drawing the letter U with a score from 0 to 10. This score was only used for evaluation and is the metric used to plot the drawing plot in A.2.

B Real-Human experiments

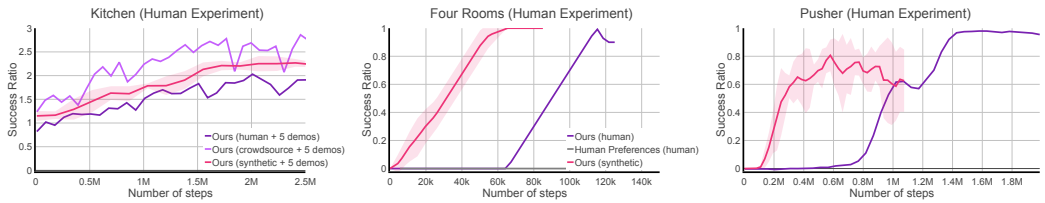

Figure B.3: Learning Progress with Human-in-the-Loop Feedback for the kitchen (**left**), four rooms navigation (**middle**) and pusher walls (**right**) for which we collected 1600, 660 and 2780 labels respectively.

In this section, we give more details on how we ran the human experiment. We designed a simple interface shown in Figure B.4. 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.

In Figure B.3 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.

Figure B.4: Screenshot of the interface from our proposed crowdsourcing platform. It shows a comparison of two image states of the kitchen environment, and the user needs to click one of the three buttons below depending on their answer of which one is best: Left/I don't know/right

B.1 Details about the crowdsourcing experiment

Subject 109 subjects participated in this pilot crowdsourcing study. Subjects were recruited from the acquaintances of the collaborators. The average time to complete the study was about 2 minutes. The subjects participated voluntarily without financial remuneration. The participants age ranged from 18 to 65+ years old. Gender Male=58.7%, Female=39.4%, Non-binary=1.8%. The participants were from 21 nationalities and participated from 13 distinct countries. More detailed information is presented in tables B.1 and B.2. There is no reason to believe that subjects experienced any physical or mental risks in the course of these studies.

Procedure This study was approved by the Institutional Review Board of [INSTITUTIONNAME] protocol E-4967.

We provide the participants with the following detailed instructions:

> Thank you very much for participating in this study. It should not take you more than a couple of minutes to complete. First of all click on the link we sent you to get directed to the main page (B.4), you can either use your phone or your computer. Please, start by filling out the form for us to get an overview of the participants' demographic. The task consists of controlling a robot to do different things in the kitchen: 1) open the slider on the right, 2) open the microwave on the right 3) open the hinge cabinet on the left. [We show a video of a successful trajectory]. To help us, we will present you two images and you need to tell us which one of the two images is closer to achieving the task. Click on the left/right button depending on whether the left/right image is better. If at some point you don't know which one is best please click the "I don't know" button. [We present a couple of examples demonstrating this]. We will show 30 pairs of images and after that, you will receive a message saying you completed the task. You can stop at any moment before that if you want.

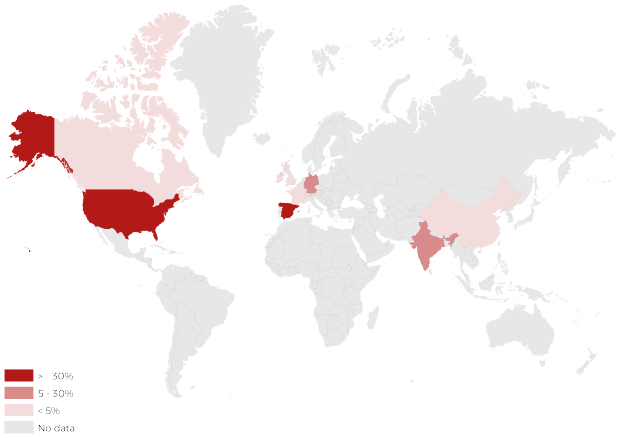

Figure B.5: Heatmap on the country representation during our crowdsourcing experiment.

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

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 B.2: Demographics on the participants of the crowdsourced data collection experiment

C Simulated Benchmarks

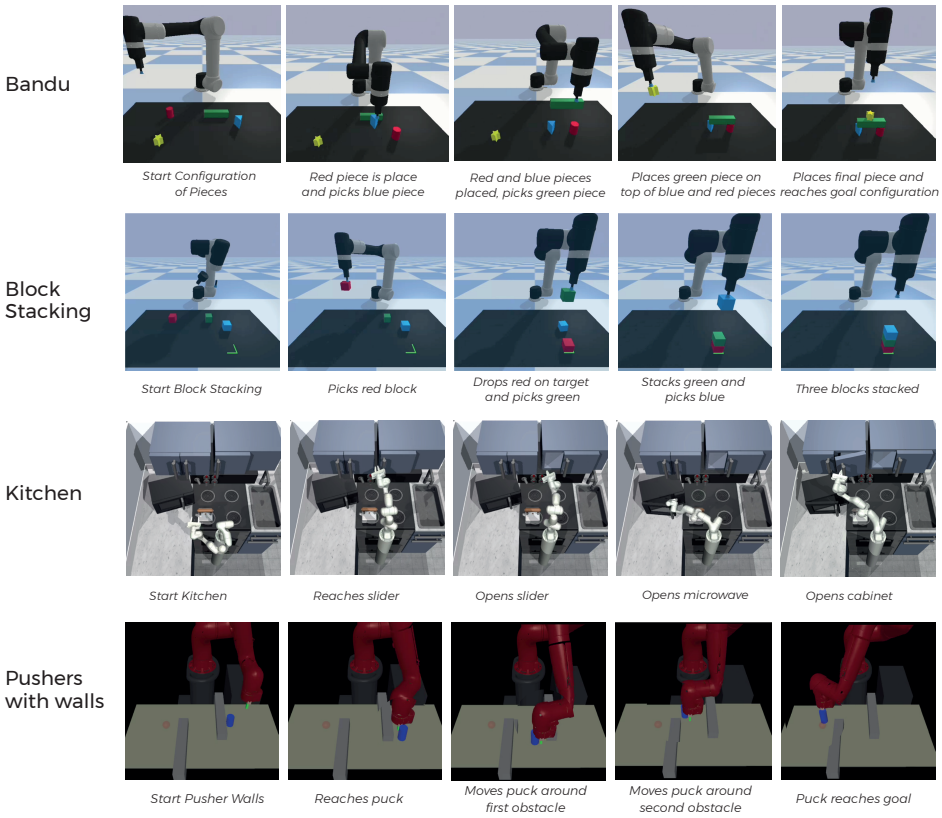

Figure C.6: Results of our method on four of the hardest benchmarks. From left to right, the timestep in the trajectory increases.

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 navigation problem in a four rooms environment, shown in Fig C.7. 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 [21].
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 F.4). 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.

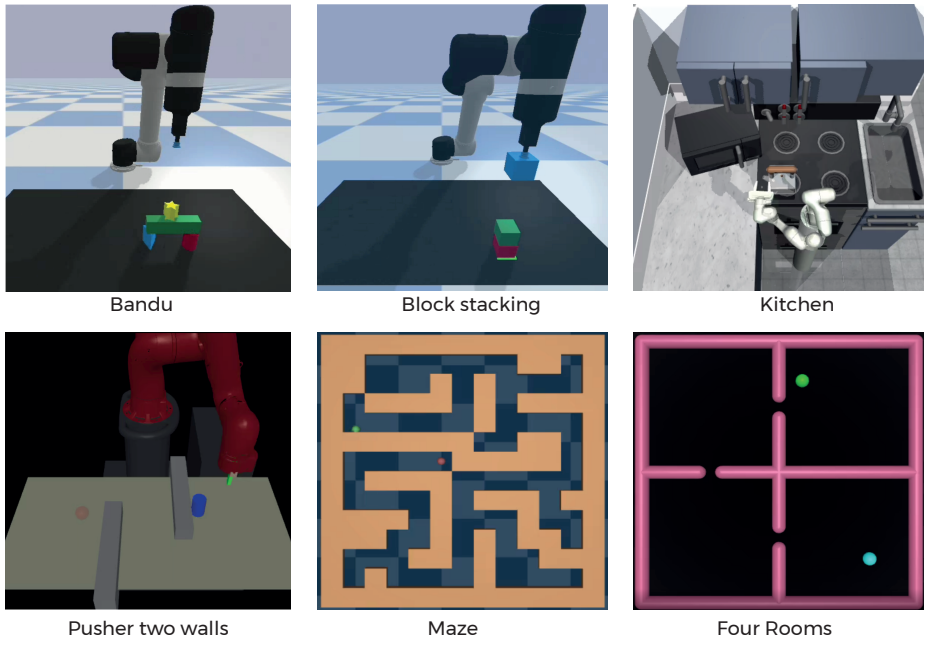

Figure C.7: Results of our method on four of the hardest benchmarks. From left to right, the timestep in the trajectory increases.

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 C.7. 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 [21]. 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:

$$r = \begin{cases} -\text{distance(arm, hinge cabinet)} - |\text{hinge cabinet target joint - hinge cabinet current joint}| \text{ , if slide cabinet and microwave opened} \\ -\text{distance(arm, microwave hinge)} - |\text{microwave target joint - microwave current joint}| - \text{bonus , if slide cabinet opened} \\ -\text{distance(arm, slide cabinet hinge)} - |\text{slide cabinet target joint - slide cabinet current joint}| - 2\text{bonus , otherwise} \end{cases} \tag{5}$$

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:

$$r = \begin{cases} -\text{distance(arm, blue block) - distance(blue block, target goal)} \text{ , if red and green block at position} \\ -\text{distance(arm, green block) - distance(green block, target goal)} - \text{bonus} \text{ , if red block at position} \\ -\text{distance(arm, red block) - distance(red block, target goal)} - 2\text{bonus} \text{ , otherwise} \end{cases} \tag{6}$$

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 C.7. The reward function is the following:

$$r = \begin{cases} -\text{distance(arm, yellow star) - distance(yellow star, target yellow star)} \text{ , if all except star at position} \\ -\text{distance(arm, green block) - distance(blue green block, target green block)} - \text{bonus} \text{ , if red and blue blocks at position} \\ -\text{distance(arm, blue triangle) - distance(blue triangle, target blue triangle)} - 2\text{bonus} \text{ , if red cylinder at position} \\ -\text{distance(arm, red cylinder) - distance(red cylinder, target red cylinder)} - 3\text{bonus} \text{ , otherwise} \end{cases} \tag{7}$$

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

D Baselines

We compare HUGE to relevant baselines from prior work.

1. **GCSL:** We compare with the iterative supervised learning algorithm for goal-reaching introduced in [21], consisting of hindsight relabeling without additional exploration.
2. **Learning from Human Preferences:** We consider the technique introduced in [12], which learns a goal-agnostic reward model using binary cross-entropy. This learned reward is then combined with an on-policy RL algorithm [43] to learn the policy.
3. **DDL:** Dynamical Distance Learning [24] 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 [43] 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.

These baselines are chosen to compare HUGE 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.

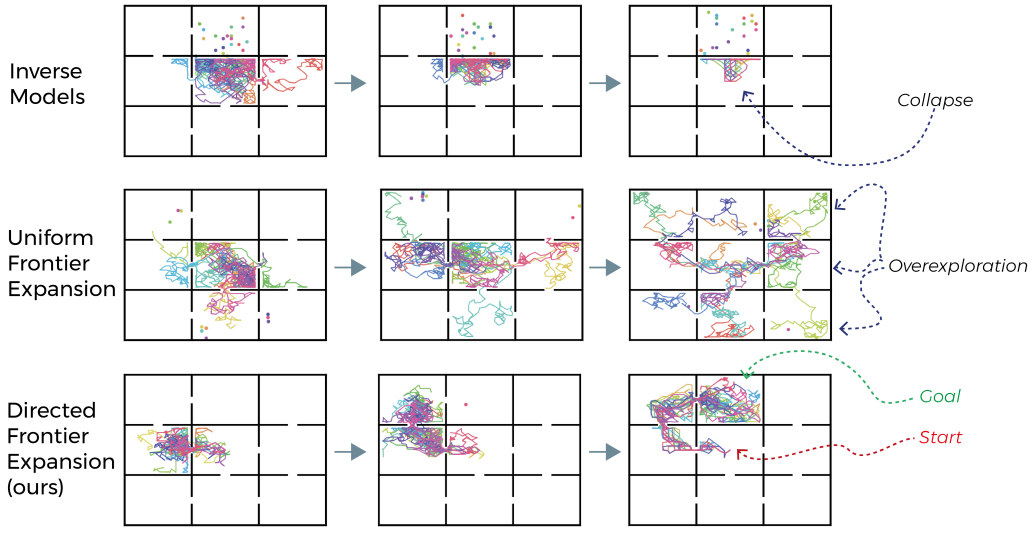

Figure D.8: Failure modes of exploration algorithms for goal-reaching. Inverse models (top) collapses and does not discover the target room (second room at the top). Uniform frontier expansion (middle) does reach the target room, but to get there it visits all possible rooms, since exploration is indiscriminate. 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.

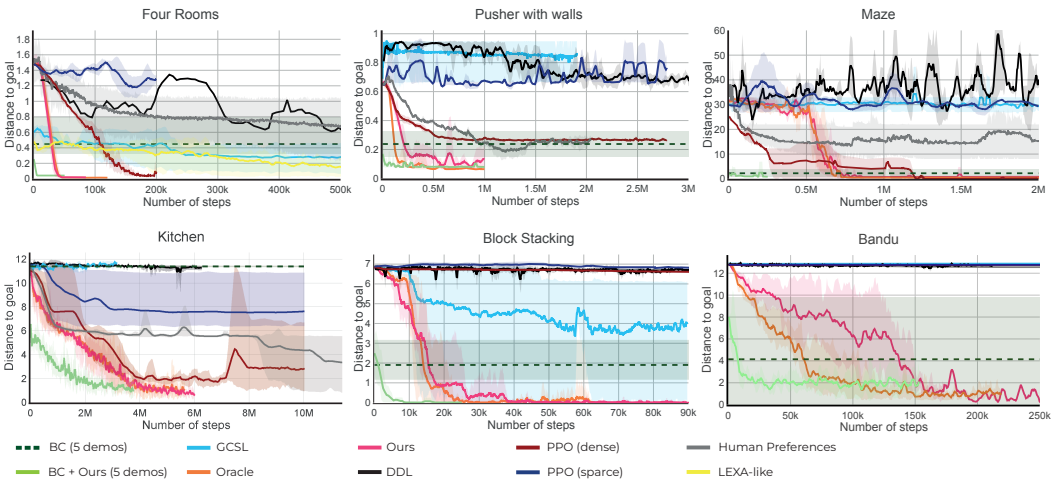

Figure D.9: Distance to the goal for each method on different benchmarks. We note that the LEXA-like exploration strategy was only implemented on the four rooms benchmark.

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 [21] (as described in Section 3) and reinforcement learning with density based exploration [34]. Exploration in GCSL relies on generalization of the policy across goals, while density based exploration rewards exploring the most novel states. We show these algorithms can fail in different ways for a simple maze environment shown in Fig D.8, where the agent starts in the middle room and must reach goals commanded in the top middle room.

As shown in Fig D.8, 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 D.8.

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 D.8, 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 D.8 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.

D.1 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 D.9 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 C.

For example, by looking at Figure D.9, 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.

Benchmark	Oracle	Ours	GCSL	Human Preferences	DDL	PPO (sparse)	PPO (dense)	LEXA style
4 rooms	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.02 \pm 0.00}$	1.15 ± 0.67	0.48 ± 0.39	0.45 ± 0.28	1.45 ± 0.13	$\mathbf{0.05 \pm 0.02}$	0.13 ± 0.18
Maze	$\mathbf{0.4 \pm 0.3}$	$\mathbf{0.8 \pm 0.3}$	29.6 ± 2.2	18.5 ± 5.6	8.54 ± 10.6	30.4 ± 0.7	$\mathbf{0.0 \pm 0.2}$	-
Pusher	$\mathbf{0.06 \pm 0.00}$	$\mathbf{0.11 \pm 0.04}$	0.85 ± 0.11	0.26 ± 0.03	0.69 ± 0.06	0.72 ± 0.06	0.27 ± 0.00	-
Kitchen	$\mathbf{1.06 \pm 0.32}$	$\mathbf{0.67 \pm 0.21}$	11.72 ± 0.11	3.43 ± 4.37	11.28 ± 0.02	7.63 ± 4.96	2.84 ± 2.72	-
Stacking	$\mathbf{0.1 \pm 0.2}$	$\mathbf{0.0 \pm 0.0}$	4.1 ± 2.3	6.5 ± 0.1	6.6 ± 0.1	6.7 ± 0.0	6.6 ± 0.0	-
Bandu	1.00 ± 0.53	$\mathbf{0.36 \pm 0.73}$	12.87 ± 0.01	12.54 ± 0.01	12.63 ± 0.21	12.75 ± 0.01	12.75 ± 0.01	-

Benchmark	BC (5 demos)	BC + Ours (5 demos)
4 rooms	0.45 ± 0.46	0.04 ± 0.00
Maze	2.25 ± 1.51	0.87 ± 1.02
Pusher	0.25 ± 0.09	0.08 ± 0.01
Kitchen	11.38 ± 0.00	0.87 ± 1.02
Stacking	1.91 ± 1.02	0.01 ± 0.00
Bandu	4.21 ± 5.47	1.87 ± 0.4

Figure D.10: 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.

E Further Analysis and Ablations

E.1 Analysis on learning from comparisons

There is a tradeoff between the frequency of labelling and the speed for the policy to converge. In Figure E.11, (**left**) we observe that if we query more frequently, the policy needs more labels to succeed, however, we also observe (**right**) that when querying less frequently it takes more timesteps to succeed. Meaning that if we provide labels more frequently, the policy is going to converge faster to the optimal policy, but will come at the cost of needing more human annotations. On the other hand, if the human annotators provide labels less frequently, it will take longer for the policy to converge to the optimal policy. The query frequency will hence be an important parameter to look into depending on what we want to optimize for, number of human labels or timesteps to succeed. We believe that for simulation experiments, we might want to optimize for using less human labels since the policy rollouts can be done very fast. However, when working with learning on the real robot, we might prefer to have humans label more frequently and reduce the number of rollouts in the real world, which is usually the bottleneck.

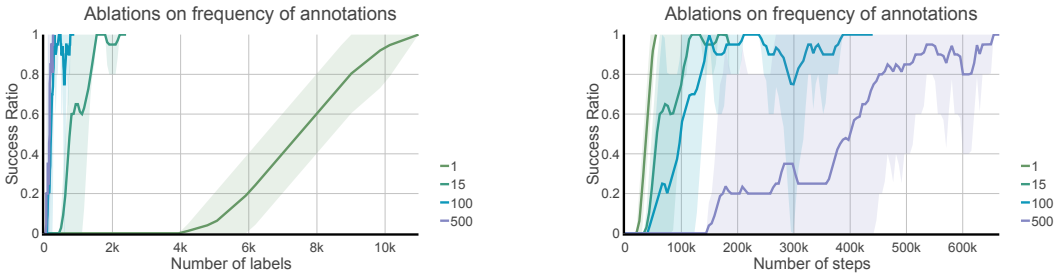

Figure E.11: On the left/right we show the number of labels/timesteps needed to succeed when varying the query frequency. 1, 15, 100, and 500 are the number of episodes between each period of querying the human for annotations. We observe a clear tradeoff between needing fewer labels to succeed against needing more timesteps. Meaning that if we query more frequently, we will need fewer timesteps to succeed and vice versa. These experiments are done in the Four Rooms benchmark.

Querying a few samples per batch is enough. In Figure E.12, (**left**) we observe that providing more labels every time we query the human leads to needing more labels to have successful policies, as expected. In **right**, however, we observe that the number of timesteps needed to have a successful policy is very similar when querying for 5,20 or 100 annotations, however, when only querying for 1 the performance drops significantly. This means that 5 labels are already enough to learn how to expand the frontier, and querying more than 5 labels brings useless information.

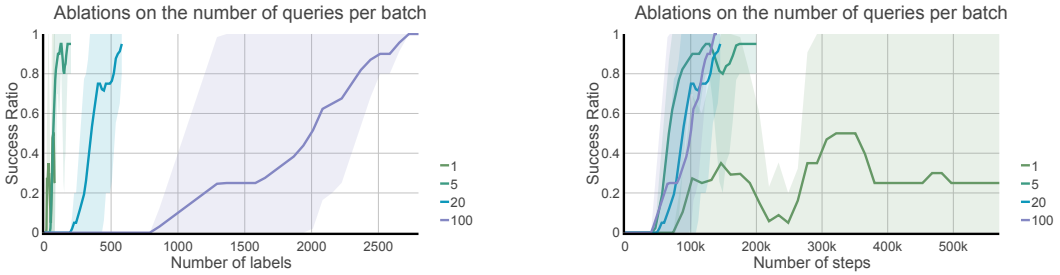

Figure E.12: On the right we show the number of steps needed to succeed in the four rooms benchmark depending on the number of comparisons queried per batch. On the left, we show the number of labels needed to succeed, again depending on the query batch size. We observe that we can go as low as 5 queries per batch, and the performance is similar to 20 and 100. Showing that too many queries bring duplicated information to the goal selector training. Also, we see that providing 1 label is not enough, degrading the performance significantly. These experiments are done in the Four Rooms benchmark.

HUGE 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 E.14. 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 E.13 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 E.18. However, HUGE still successfully reaches the goal. As we can see, there are 3 modes in the final step (4th subfigure in E.13). 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.

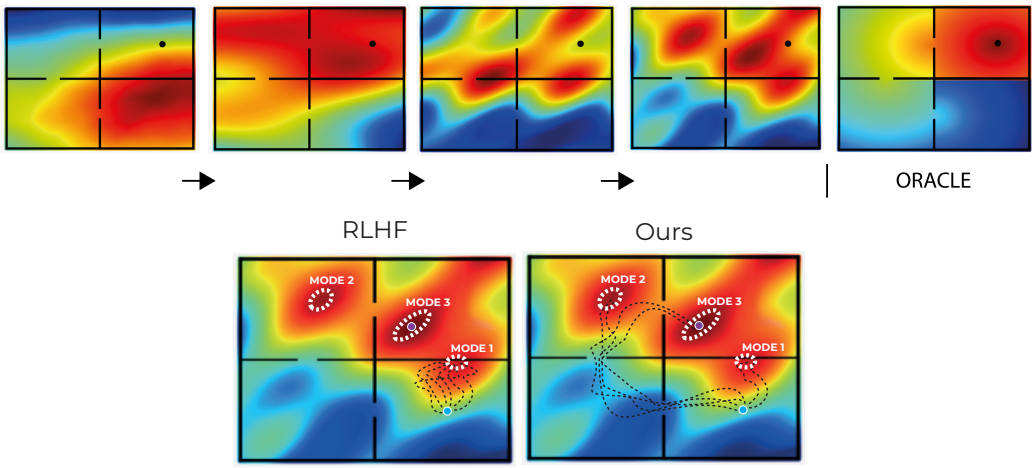

Figure E.13: Evolution of the learned goal selector when the distance for the synthetic human has a noise of 1. We observe that the goal selector is not accurate, however, our method still successfully reaches the goal, hence, it is robust to inaccurate goal selectors. This would not be the case for methods that use this goal selector as a reward function to run model-free RL, due to the noise on it and multiple local minimas and maximas.

HUGE is robust to underlying simple reward functions. In Figure E.15 we show 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.

HUGE can learn when no labels are provided. This property of HUGE 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 [24], which will not have this advantage. From Figure E.18 we observe that a parametric goal selector has the capacity to generalize while, by definition a non-parametric goal selection [24] will not. Thereafter, using a parametric reward model that has non-degenerate extrapolation can lead to significantly more frontier expansion. In Figure E.16 we show how our method succeeds in reaching the final goal room even if the goal selector has stopped training when the agent enters any of the previous rooms. However, this comes at a cost in much slower convergence.

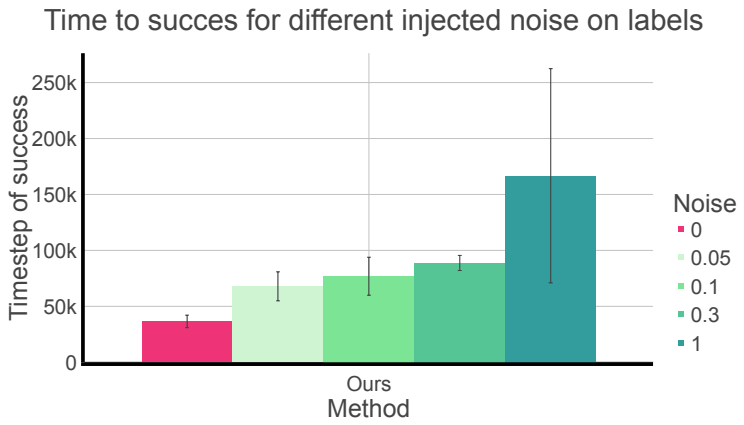

Figure E.14: Show the effect of adding Gaussian noise in the labels provided by the human on the Four Rooms benchmark. We observe that our method is robust to different amounts of added noise, however, as noise increases, so will the timesteps needed to succeed. Noise is injected into the distance function used by the synthetic human to provide labels, which means that with higher noise the probability of the comparison being wrong will increase. For context, the distance between the initial state and the goal is around 1.6.

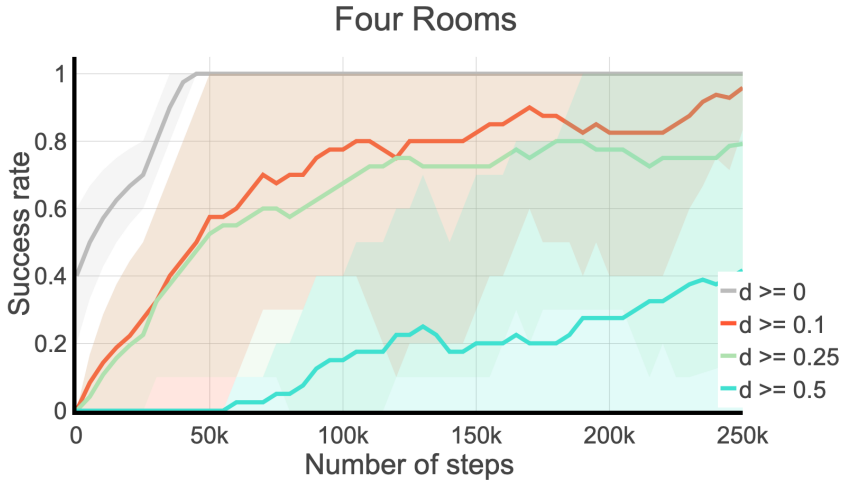

Figure E.15: Comparison on the effect of simplified reward functions providing the synthetic human annotations.

E.2 Goal selector Analysis

Learning a goal selector is more feedback efficient than directly using the human feedback. In figure E.17 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 frequencies 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.

In figure E.18, we show the goal selector will have non-trivial generalization, allowing us to continue expanding the frontier even when no human is present.

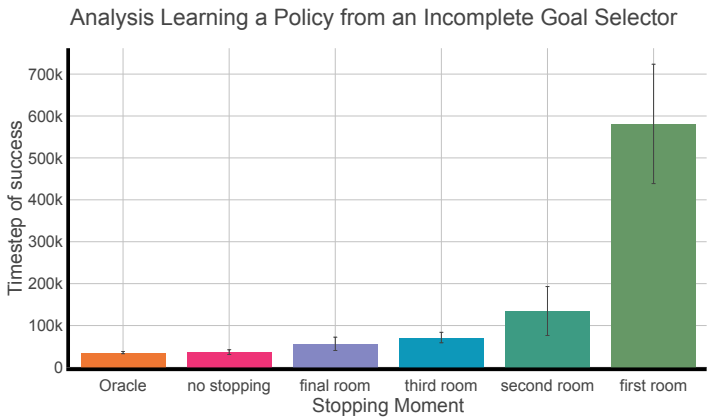

Figure E.16: Effect of freezing the goal selector at different points in the learning of the policy on how long it takes to learn a successful policy on the Four Rooms benchmark. We see that an earlier stop in the training leads to an increase in the timesteps needed to succeed. However, even if we stop 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.

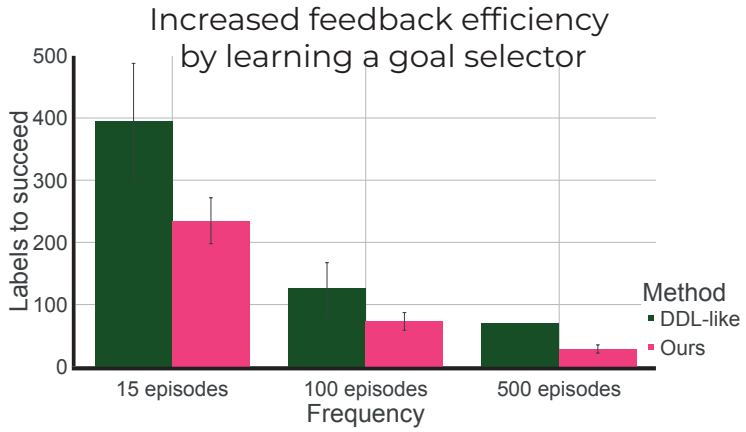

Figure E.17: Comparison of the number of labels needed to succeed when using a parametric goal selector (Ours) against directly using the goal selected by the human (DDL).

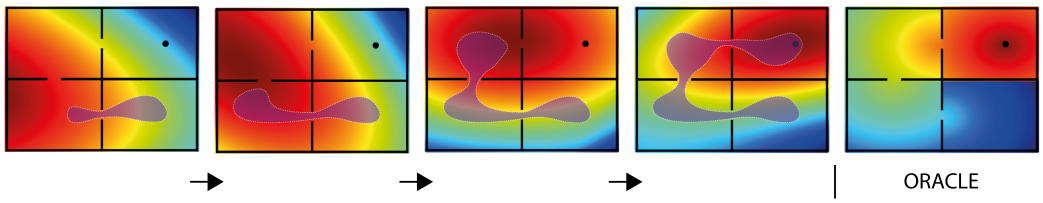

Figure E.18: Progress of goal selector learning in the four rooms environment as learning progresses it gets closer to the target (oracle on the right). The purple area represents the visited states by the agent at that point. We observe that the goal selector provides extrapolation which will help the training with fewer annotations.

Furthermore, in E.19 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.

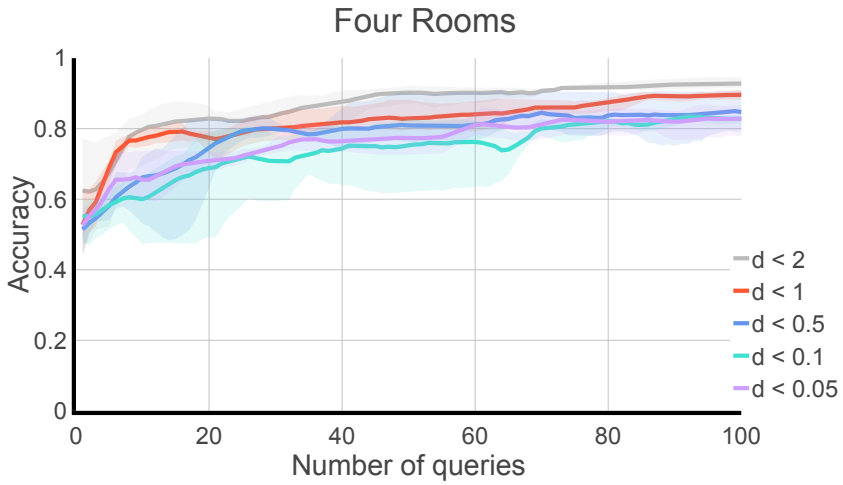

Figure E.19: Accuracy of the goal selector depending on the number of queries and dependent on the distance d between the states compared in the labels.

Qualitative analysis of the generalization of the goal selector. In this qualitative analysis, we show visualizations of the learned goal selector as different rooms are discovered during the learning process in the four-rooms domain Fig E.20. The goal selector model shows nontrivial extrapolation and can potentially provide guidance even beyond the set of states it is trained on.

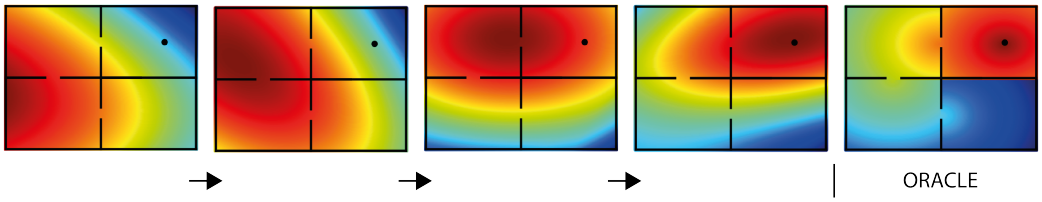

Figure E.20: The goal selector learns and converges to a result close to the oracle (rightmost) as epochs increase (left to right). We observe how this goal selector gets updated iteratively as the frontier expands. Colder colors mean a lower reward for that state, whereas warmer colors mean a higher reward for that state, in this case, this is equivalent to the distance to the goal.

E.3 Compatibility of HUGE with Learning from Trajectory Demonstrations

As we mention in Section 4.4, HUGE is compatible with learning from trajectory demonstrations. In Figure E.21, we show how HUGE can improve the performance of simple imitation learning starting from different amounts of demonstrations. Given the number of demonstrations, imitation learning fails on less than 10 demonstrations, and with HUGE we can improve the policy to succeed in all cases.

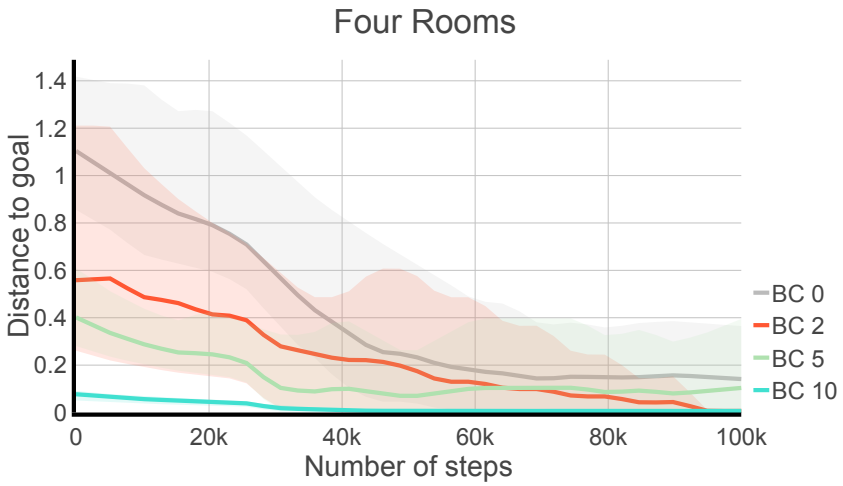

Figure E.21: The figure depicts the distance to the goal in the Four Rooms environment when using a policy pre-trained via Behaviour Cloning with 0, 2, 5, and 10 demonstrations, respectively. We see that using BC on a small number of demonstrations helps to boost the performance of our method. Also, 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.

E.4 Analysis on implementation details

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 E.22 we see the resulting performance with and without taking out redundant steps. In 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.

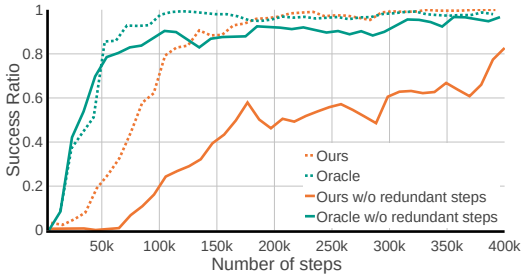

Figure E.22: Effect of the use of exploration after reaching the commanded goal on the performance

F Implementation Details

For training the models and running the experiments, we had access to several workstations with one GeForce RTX 2080 Ti or one GeForce RTX 3090. It took on average 8 hours on these machines to run 4 seeds for each one of the baselines and our method. We account the total amount of compute hours would be around 1440 hours for the whole project, taking into account, experimentation and testing the algorithms.

F.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 [48]. For this reason, we used them in some of our experiments, as detailed in Section F. More precisely, when used, we added an additional layer with Fourier features of size 40 times the input dimension.

F.2 Training details

The details of the parameters with which the results have been obtained will be disclosed in this section. In particular, Table F.4 depicts the parameters used for the different benchmarks, while Table F.3 contains the hyperparameter configuration used for the different algorithms.

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 [43] 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 [43] plus	
Buffer size	8192
Policy architecture	MLP $(400, 600, 600, 300)$

Table F.3: Hyperparameters setting for the algorithms

Environment	Four rooms	Maze	Pushing around Obstacles	Kitchen	Block Stacking	Bandu
Steps per trajectory	50	250	100	100	10	12
Label from last k steps	10	50	10	20	10	12

Table F.4: Benchmark-related parameters

References

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

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

[3] M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In D. Precup and Y. W. Teh, editors, *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*, pages 263–272. PMLR, 2017.

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

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

[6] E. Biyik and D. Sadigh. 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*, pages 519–528. PMLR, 2018.

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

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

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

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

[11] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

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

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

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

[15] Y. Cui, P. Koppol, H. Admoni, S. Niekum, R. G. Simmons, A. Steinfeld, and T. Fitzgerald. Understanding the relationship between interactions and outcomes in human-in-the-loop machine learning. In Z. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4382–4391. ijcai.org, 2021.

[16] T. Davchev, O. O. Sushkov, J. Regli, S. Schaal, Y. Aytar, M. Wulfmeier, and J. Scholz. 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, 2022.

[17] R. Devidze, P. Kamalaruban, and A. Singla. Exploration-guided reward shaping for reinforcement learning under sparse rewards. *Advances in Neural Information Processing Systems*, 35:5829–5842, 2022.

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

[19] B. Eysenbach, R. Salakhutdinov, and S. Levine. 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, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15220–15231, 2019.

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

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

[22] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, pages 1025–1037. PMLR, 2019.

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

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

[25] A. Jain, D. Das, and A. Saxena. Planit: A crowdsourcing approach for learning to plan paths from large scale preference feedback. *CoRR*, abs/1406.2616, 2014.

[26] L. P. Kaelbling. Learning to achieve goals. Citeseer.

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

[28] K. Lee, L. Smith, and P. Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, 2021.

[29] K. Lee, L. M. Smith, and P. Abbeel. PEBBLE: feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In M. Meila and T. Zhang, editors, *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*, pages 6152–6163. PMLR, 2021.

[30] A. Levy, G. D. Konidaris, R. P. Jr., and K. Saenko. 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, 2019.

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

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

[33] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *CoRR*, abs/2210.06407, 2022.

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

[35] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[36] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[37] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. 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*, pages 2146–2153. IEEE, 2017.

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

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

[40] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022.

[41] V. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. 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*, pages 7783–7792. PMLR, 2020.

[42] M. A. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. V. de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing - solving sparse reward tasks from scratch. *CoRR*, abs/1802.10567, 2018.

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

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

[45] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2753–2762, 2017.

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

[47] G. Warnell, N. R. Waytowich, V. Lawhern, and P. Stone. Deep TAMER: interactive agent shaping in high-dimensional state spaces. In *AAAI*, 2018.

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