Discovering Generalizable Spatial Goal Representations via Graph-based Active Reward Learning

Aviv Netanyahu [*1] **Tianmin Shu** [*1 2] **Joshua Tenenbaum** [1 2] **Pulkit Agrawal** [1]

Abstract

In this work, we consider one-shot imitation learning for object rearrangement tasks, where an AI agent needs to watch a single expert demonstration and learn to perform the same task in different environments. To achieve a strong generalization, the AI agent must infer the spatial goal specification for the task. However, there can be multiple goal specifications that fit the given demonstration. To address this, we propose a reward learning approach, Graph-based Equivalence Mappings (GEM), that can discover spatial goal representations that are aligned with the intended goal specification, enabling successful generalization in unseen environments. Specifically, GEM represents a spatial goal specification by a reward function conditioned on i) a graph indicating important spatial relationships between objects and ii) state equivalence mappings for each edge in the graph indicating invariant properties of the corresponding relationship. GEM combines inverse reinforcement learning and active reward learning to efficiently improve the reward function by utilizing the graph structure and domain randomization enabled by the equivalence mappings. We conducted experiments with simulated oracles and with human subjects. The results show that GEM can drastically improve the generalizability of the learned goal representations over strong baselines.[1]

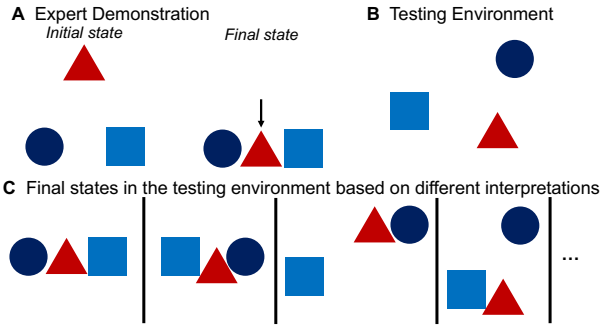

Figure 1. Illustration of our problem setup. We first (**A**) show a single expert demonstration for an object rearrangement task to an agent, and then (**B**) ask the agent to reach the same goal in unseen testing environments. (**C**) Multiple spatial goals can interpret the expert demonstration, each leading to a distinct task execution in the testing environments. For instance, from left to right, the four possible spatial goals shown here are i) triangle is to the right of circle and to the left of square; ii) triangle is close to circle and square; iii) triangle is to the left of circle; iv) triangle is close to square.

1. Introduction

To build AI agents that can assist humans in real world settings, we have to first enable them to learn to perform any new tasks defined by a human user. To achieve this, an AI agent has to acquire two types of key abilities: i) the ability to develop a generalizable understanding of the goal or task specification intended by the human user and ii) the ability to plan or learn a policy for a given goal. In this work, we aim at engineering the first key ability for an AI agent. In particular, we focus on object rearrangement tasks, where an agent must reason about the spatial goals that define a set of desired spatial relationships between objects. For instance, to set up a dinner table, one has to know how to place the plates and the utensils appropriately. These tasks are commonly studied in robots (Shah et al., 2018; Yan et al., 2020; Rowe et al., 2019) and embodied AI (Puig et al., 2021; Srivastava et al., 2022), serving as as a foundation for a broader range of tasks such as house keeping and manufacturing.

[*]Equal contribution [1] Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA [2] Dept. of Brain and Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA. Correspondence to: Aviv Netanyahu <avivn@mit.edu>, Tianmin Shu <tshu@mit.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]Project website: https://www.tshu.io/GEM

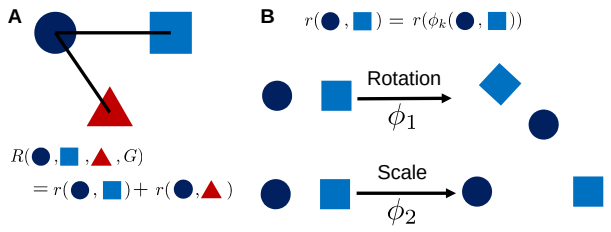

Figure 2. (**A**) We use a compositional reward function, R, conditioned on a graph, G, as the spatial goal representation. (**B**) For each edge, we may apply certain state equivalence mappings for improving representation learning to achieve a better generalization beyond the expert demonstration. Each type of mapping indicates a type of invariance of the intended spatial relationship between a pair of objects. Thus the reward for the edge would maintain the same after applying the state mapping. Specifically, for the rotation-invariant mapping ϕ_1, the relative orientation between objects can be randomized while the reward remains the same; for the scale-invariant mapping ϕ_2, the change in the distance between objects does not affect the reward.

One way to train agents to perform a new object rearrangement task is to provide manual goal specification, such as detailed instructions or hand-craft reward functions. However, creating a manual definition for the goal requires expert knowledge, and inaccurate definitions may cause misspecification. Instead, our work focuses on a more general paradigm, i.e., one-shot imitation learning, where the agent watches the human user performing the task once (Figure 1**A**) and learns to perform the same task in unseen environments (Figure 1**B**).

While one-shot imitation learning is a convenient paradigm for teaching new tasks to agents, it is also extremely challenging due to the fact that there could be multiple goal specifications that explain the given expert demonstration well. For instance, consider the task illustrated in Figure 1. There are multiple interpretations of the intended goal spatial relationships based on the demonstration, each of which will lead to a different task execution in a new environment (Figure 1**C**). Without a correct understanding of the true goal, an agent cannot successfully perform the task.

To address this challenge, our work improves both i) the representation of spatial goal specification and ii) the acquisition of such representation that can reveal the true spatial goal.

First, we represent the intended spatial goal by a compositional reward function conditioned on a sparse graph (Figure 2**A**) where the graph indicates *whether* there is an important spatial relationship between a pair of objects and each edge has a reward function implicitly describing *what* the intended spatial relationship is between the corresponding pair of objects. Unlike prior work on reasoning about spatial relationships and graphical representations of goals, we do not classify a relation out of a manually defined set of predicates (e.g., close, above), but intend to discover those predicates through the graph and the reward components for the edges in the graph implicitly.

Second, we propose a novel reward learning algorithm, Graph-based Equivalence Mappings (GEM), connecting offline reward learning with active reward learning. As shown in Figure 3, GEM consists of two phases. In the initial phase (Figure 3**A**), we first learn a reward function conditioned on a fully connected graph from the expert demonstration using adversarial inverse reward learning (AIRL) (Fu et al., 2018) with a model-based extension, i.e., M-AIRL. This reward function is guaranteed to provide a good fit on the states in the expert demonstration, which offers us a set of good state-reward pairs as a training set. However, this set is only limited to the training situation, thus the initial reward function learned from M-AIRL may not generalize well to unseen states during test. To acquire new state-reward pairs that are not covered by the expert demonstration and improve the spatial goal representations to match with the newly acquired data, we then conduct active reward refinement (Figure 3**B**) following the initial training.

Prior works on active reward learning typically collect new state-reward training data purely based on the states generated in the queries shown to the oracle and the feedback received from the oracle. When faced with a large state space, this paradigm often requires a large amount of queries to acquire sufficient training data. To overcome this, we aim to augment the existing training states by randomizing each state in the existing set without changing the corresponding reward. This can be achieved by applying state equivalence mappings (Figure 2**B**) to edges, which is a type of state transformation that identifies equivalent states. A similar idea has been previously applied to multi-agent policy learning for improving zero-shot generalization (Hu et al., 2020). Here, each type of equivalence mapping indicates a type of invariance for the intended spatial representation between a pair of objects. Thus, when applying valid equivalence mappings, we can synthesize new state-reward pairs with known rewards without the need to acquire oracle feedback on each new state. Following this intuition, our active reward refinement iteratively proposes a new hypothesis including a new graph and new state equivalence mappings assigned to the edges in the graph, finetunes the reward function based on the new graph and the augmented training set, and generates informative queries for the oracle to verify the hypothesis. For instance, in Figure 3**B**, the edge between the square and the circle is removed in the new graph, while the rotation-invariant mapping is preserved for the remaining edge. Consequently, the sampled query moves the square away and rotates the triangle around the circle, so that we can verify whether the removed edge is

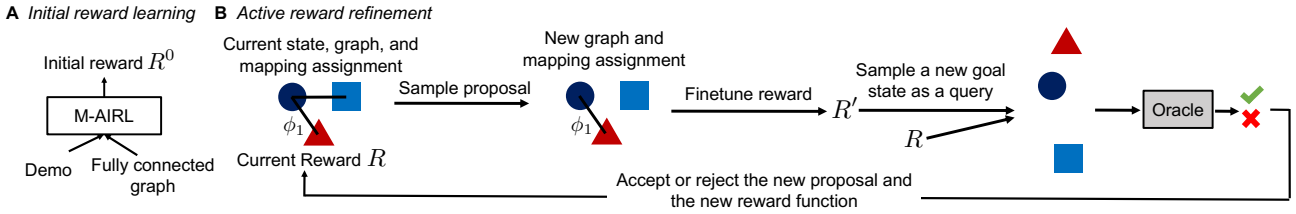

Figure 3. Overview of GEM. The reward learning consists of two phases, offering a novel connection between (**A**) model-based inverse reinforcement learning that predicts an initial reward and (**B**) active reward learning. Given an expert demonstration, we first initialize the reward function conditioned on a fully connected graph (i.e., R^0) using model-based adversarial inverse reinforcement learning (M-AIRL). R^0 provides a good estimation of expert demonstration state rewards with a theoretical fitness guarantee. To improve generalization beyond states in the expert demonstration, we update the reward function iteratively. At each iteration we propose a new graph or a new equivalence mapping assignment for the edges in the new graph. We then finetune the reward function conditioned on the new graph using data augmented by the new equivalence mappings. To verify the fitness of the proposed graph and the equivalence mappings, we generate a new goal state that can differentiate the new reward R' from the current reward R as an informative query for the oracle. Based on the oracle feedback (i.e., whether the new goal state is acceptable), we update the current proposal, reward function, and state accordingly. We also collect the query states as additional training data for the reward finetuning at future iterations.

important and whether the rotation-invariance holds for the connected edge through this query.

We conducted experiments with a simulated oracle and with human subjects in a 2D physics simulation environment, Watch&Move. In each task, the goal is to move the objects to satisfy spatial relationships intended by the oracle. We compared GEM against recent baselines for imitation learning and active reward learning, and found that GEM significantly outperformed the baselines, both in terms of the generalizability and the sample efficiency (i.e., less oracle queries).

In summary, our main contributions are: i) a generalizable spatial goal representation using a compositional reward function conditioned on a graph and state equivalence mappings, ii) a novel reward learning algorithm, GEM, for discovering spatial goal representations by connecting inverse reinforcement learning with sample-efficient active reward learning, and iii) a new physics simulation environment, Watch&Move, for evaluating one-shot imitation learning approaches, with a focus on generalization.

2. Related Work

Goal inference. One of the key aspects of the work is to reason about the goal for a task based on the expert's plan. There has been rich history in goal inference building socially intelligent AI systems (Baker et al., 2017; Puig et al., 2021; Netanyahu et al., 2021; Shah et al., 2018; Yan et al., 2020). However, prior work on goal inference typically assumed a limited goal space such as a set of discrete goals (Baker et al., 2017), a finite set of predicates (Puig et al., 2021; Netanyahu et al., 2021; Shah et al., 2018), or a target location (Cao et al., 2020). These assumptions greatly limited the kinds of goals a system can infer. In contrast, our approach can discover generalizable spatial goal representations with much less restriction for the spatial goal specification.

Inverse reinforcement learning. Most existing imitation learning approaches can be categorized into behavioral cloning (BC) and inverse reinforcement learning (IRL) (Ghasemipour et al., 2020). These two types of methodologies provide two distinct learning objectives – BC aims to directly mimic the expert policy, whereas IRL attempts to recover the reward function that could produce the expert policy. Intuitively, learning a reward can achieve better generalization in novel environments since the learned reward function may still be valid in the new environment, whereas a policy inferred by demonstrations may no longer be suitable when the environment distribution changes (covariate shift) (Shimodaira, 2000). The main challenge in IRL is that there are usually multiple rewards that can explain the expert demonstrations (Ng et al., 2000), especially with limited demonstrations. Policies learned from IRL simultaneously with the reward are guaranteed to only perform well on the expert distribution (Ghasemipour et al., 2020). State-only AIRL (Fu et al., 2018) is guaranteed to learn a reward disentangled from environment dynamics, but may also suffer from covariate shift. Hence, we propose a new learning paradigm that extends IRL with graph-based active reward learning.

One-shot imitation learning. Imitation learning has long been a subject of interest (Schaal, 1999; Nehaniv et al., 2002; Abbeel & Ng, 2004; Billard et al., 2004; Argall et al., 2009). Specifically, there has been work on one-shot imitation learning (Duan et al., 2017; Bonardi et al., 2020; Huang et al., 2019; Yu et al., 2018), which often adopted

a meta-learning framework, where the objective is to learn how to learn from a single demonstration through training with a distribution of tasks. This can be done either through meta policy learning (Finn et al., 2017) or meta reward learning (Xu et al., 2019). However, existing works focused on tasks that have a high similarity, e.g., pushing an object to different locations (Finn et al., 2017). They have also not addressed generalizibility of the learned policies or reward functions to unseen environments. There are two main challenges for achieving a strong generalization in one-shot imitation learning: it is hard i) to learn to reach a given goal in unseen environments and ii) to infer the true goal from a single demonstration. In this work, we focus on the second challenge, with the assumption of having access to a world model and a planner that can reach any physically plausible goal state. We believe this is a first step towards engineering a generalizable one-shot imitation learning system.

Policy/reward learning from human feedback. In addition to learning from demonstrations, prior work has also proposed methods for learning policies (Ross et al., 2011; Griffith et al., 2013; Loftin et al., 2014; MacGlashan et al., 2017; Arumugam et al., 2019; Wang et al., 2022) or reward functions (Daniel et al., 2014; 2015; Su et al., 2016; Bıyık et al., 2019; Cui & Niekum, 2018; Brown et al., 2019; Reddy et al., 2020) from human feedback. This can be achieved through queries that ask for human preferences (Christiano et al., 2017; Brown et al., 2019) or a direct evaluation (reward) on states (Ross et al., 2011; Reddy et al., 2020). Inspired by this, our approach also uses an active reward learning scheme to improve the reward function from oracle feedback. However, when the state and action spaces are large, it is difficult to obtain sufficient amount of data from a small number of queries. In this work, we aim to address this by better utilizing the limited queries and oracle feedback. Specifically, instead of only generating trajectories or states for the queries, we propose generalizable goal representations and verify them through smart query generation.

3. Preliminaries

Our initial learning phase adopts adversarial inverse reinforcement learning (AIRL) (Fu et al., 2018), which was proposed to achieve robust reward generalization for unseen dynamics. We briefly introduce AIRL and present a model-based extension to the original AIRL below.

3.1. Adversarial Inverse Reinforcement Learning

IRL considers an MDP process $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \rho_0 \rangle$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(\cdot|a,s)$ is the state transition distribution, $r(s,a)$ is the reward function, γ is the discount factor, and ρ_0 is the initial state distribution. The goal of IRL is to learn a reward function $r_\theta(s,a)$ that can approximate the expert policy on the given expert demonstrations $\mathcal{D} = \{\Gamma_1, \cdots, \Gamma_M\}$, where $\Gamma_i = \{(s^t, a^t)\}_{t=0}^T$ is a sequence of state and action pairs in a demonstration. It achieves this objective by maximizing the likelihood of observing the expert demonstrations given the reward function.

$$\max_\theta E_{\Gamma\sim\mathcal{D}}[\log p_\theta(\Gamma)], \tag{1}$$

where $p_\theta(\Gamma) \propto p(s_0)\prod_{t=0}^T p(s^{t+1}|s^t,a^t)e^{\gamma^t r_\theta(s^t,a^t)}$ is the likelihood of the demonstrations given the reward function. AIRL formulates this optimization as adversarial training, where it learns to approximate the advantage function for the expert policy through a discriminator. The discriminator distinguishes between generated trajectories from a learned policy $\pi(a|s)$ (as fake examples) and the expert demonsrations (as real examples):

$$D_{\theta,\omega}(s,a,s') = \frac{\exp\{f_{\theta,\omega}(s,a,s')\}}{\exp\{f_{\theta,\omega}(s,a,s')\} + \pi(a|s)}, \tag{2}$$

where $f_{\theta,\omega}$ is the advantage function consisting of an approximated reward function g_θ as well as a shaping function h_ω:

$$f_{\theta,\omega} = g_\theta(s,a) + \gamma h_\omega(s') - h_\omega(s). \tag{3}$$

When the reward function only depends on state s, AIRL can guarantee that the learned reward function g_θ and the shaping function h_ω can approximate the ground-truth reward function r^* and the ground-truth value function $V^*(s)$ up to a constant respectively. However, the learned reward may not generalize well to states different from the ones shown in the expert demonstrations.

3.2. Model-based AIRL

The original AIRL uses a model-free RL training, which is often difficult in tasks with large state and action spaces (such as the multi-object rearrangement tasks studied in this work). To address this issue there has been work on model-based AIRL (Sun et al., 2021). Similarly, we propose a simple model-based extension given a world model $p(\cdot|s,a)$ learned or given by a simulator. The idea is to utilize the shaping function $h(s)$ in Eq. (3), which is guaranteed to approximate the ground-truth value function in the demonstrations, combined with one-step-ahead state predication to define the policy in Eq. (2) instead of learning a policy π using model-free RL:

$$\pi(a|s) = \frac{\exp\{\beta \sum_{s'} p(s'|a,s)h(s')\}}{\sum_{a'\in\mathcal{A}} \exp\{\beta \sum_{s'} p(s'|a',s)h(s')\}}, \tag{4}$$

where β is a constant coefficient.

4. Graph-based Equivalence Mappings

As a representation of a spatial goal, a reward function can adequately describe the fitness of a spatial configuration w.r.t. any intended goal spatial relationships including

both logical and continuous relationships. When learned properly, the trained reward function also enables a generalization of the corresponding goal in unseen physical environments, avoiding over-imitation, unlike direct policy imitation. However, without a diverse set of demonstrations, multiple reward functions may perfectly fit the expert trajectories, and it is impossible to disambiguate the true reward solely based on the given demonstration. To solve this, we propose a novel active reward learning approach, Graph-based Equivalence Mappings (GEM) that learns a compositional reward function conditioned on a sparse graph and state equivalence mappings (Figure 2). As illustrated in Figure 3, GEM consists of two learning phases, combining both model-based inverse RL and active reward learning. Starting from the initial reward function that has a theoretical fitness guarantee limited to the expert demonstration, GEM iteratively refines the reward function through proposed new graphs and state equivalence mappings and verifies the new reward function via informative queries for an oracle. In this section, we first introduce our compositional reward function and then present the two-phase learning algorithm.

4.1. Compositional Reward Function as a Spatial Goal Representation

We represent each state as $s = (x_i)_{i \in N}$, where N is a set of objects and x_i is the state of object i. We indicate the important spatial relationships for a task by a graph, $G = (N, E)$, where each edge $(i, j) \in E$ shows that the spatial relationship between object i and object j is part of the goal specification. Here, we focus on pairwise spatial relationships, but it is possible to extend this to higher-order relationships. As shown in Figure 2**A**, given the graph and the state, we define a compositional reward function as a spatial goal representation to implicitly describe the goal spatial relationships for a task:

$$R(s, G) = \frac{1}{|E|} \sum_{(i,j) \in E} r(x_i, x_j). \tag{5}$$

Spatial relationships may have certain invariance properties. For instance, relationships describing the desired distance between two objects are invariant to the rotation applied to this pair of objects. By utilizing correct invariance properties, we can transform a state seen in the training environment to a new state that has the same reward, as they represent the same spatial relationship. Essentially this process augments the data by domain randomization (Tobin et al., 2017). To model different invariance properties, we introduce a set of possible state equivalence mappings $\{\phi_k\}_{k \in K}$ as shown in Figure 2**B**, where each type of mapping ϕ_k can transform the states of a pair of objects (i, j), and ensures that the reward component for that edge does not change, i.e., $r(x_i, x_j) = r(\phi_k(x_i, x_j))$. When applying a mapping, we may randomize the invariance aspect of the state to sample a new state. For instance, for applying the rotation-invariant mapping once, we randomize the relative orientation between the two objects for the state transformation. Please refer to Appendix A.1 for more details.

We denote the mappings assignment for all the edges in a graph as $I = \{\delta_{i,j,k}\}_{(i,j) \in E, k \in K}$, where $\delta_{i,j,k}$ is a binary variable indicating whether ϕ_k can be applied to edge (i, j). The state mapping for the whole graph is then defined as $\Phi(s, I)$, where it recursively applies mappings assigned to each edge. Note that we transform the state for each edge independently so that the transformation of one edge will not affect the state of other edges that share a common node with this edge. For more details, please refer to Appendix A.1.

In this work, we consider two types of mappings illustrated in Figure 2**B** as a gentle inductive bias provided by domain knowledge, but other types of mappings can also be applicable for different domains. Critically, we only provide a set of candidate mappings but do not assume to have the knowledge of which mappings can be applied to a specific spatial relationship, unlike (Hu et al., 2020). We instead learn to assign valid mappings to each edge through active reward learning.

These equivalence mappings allow us to augment the states from the expert demonstration and those acquired through queries, creating an infinite number of states that have equivalent spatial relationships.

4.2. Two-Phase Reward Learning

4.2.1. Initial Reward Learning

Given the expert demonstration Γ, we first use M-AIRL described in Section 3 to train an initial reward function $R^0(s, G^0; \theta^0)$ conditioned on a fully connected graph G^0, where θ^0 are the parameters of the initial reward function. This initial reward function provides a good reward approximation for all the states in Γ, approaching the ground-truth with a constant offset.

4.2.2. Active Reward Refinement

To improve the generalization of the initial reward function to states beyond expert states, we then aim to discover a graph structure, G, and equivalence mapping assignment for the edges in the graph, I, through active learning, and refine the reward function based on the inferred graph and the equivalence mappings.

The active reward refinement is outlined in Algorithm 1. For the reward refinement, we consider three training sets, (1) all states in the expert demonstrations and their rewards based on R^0, $S_D = \{(s^t, r^t = R^0(s^t | G^0, \theta^0))\}_{t=0}^{T}$, (2) a positive state set initialized with the final state of the expert

Algorithm 1 Active Reward Refinement

Input: G_0, I_0, R_0, S_D, L_{max}
$S_+ \leftarrow \{s^T\}$, $S_- \leftarrow \emptyset$, $I_0 \leftarrow \emptyset$
First reach the final state of the expert demonstration, denoted as s_q^0
for $i = 1$ **to** L_{max} **do**
  $G', I' \sim Q(G^l, I^l | G^{l-1}, I^l)$
  Train θ' based on Eq. (7)
  Sample a new query state s_q' based on Eq. (8)
  Reach s_q' and get oracle feedback o
  if $o =$ acceptable **then**
    $G^l \leftarrow G'$, $I^l \leftarrow I'$, $s_q^l \leftarrow s_q'$, $\theta^l \leftarrow \theta'$
    $S_+ \leftarrow S_+ \cup \{s_q'\}$, $S_D \leftarrow (s_q', r^T)$
  else
    $G^l \leftarrow G^{l-1}$, $I^l \leftarrow I^{l-1}$, $s_q^l \leftarrow s_q^l$, $\theta^{l-1} \leftarrow \theta^{l-1}$
    $S_- \leftarrow S_- \cup \{s_q'\}$
    Move objects back to s_q^{l-1}
  end if
end for

demonstration S_+, and (3) a negative state set, S_-. By querying, we collect more states for the positive set and the negative set based on the oracle judgment. We know the corresponding reward of the states in S_D based on R_0, and assume that rewards for states in S_+ have higher rewards than any state in S_-.

The main purpose of the queries is to find equivalently good goal states that are visually different from the goal state shown in the expert demonstration. To this end, at the start of the active learning phase, we first reach the final state of the expert demonstration. We can achieve this by applying the learned approximated value function (the policy is defined in Eq. (4)).

At each iteration l, we first propose a new graph G' and a new equivalence mapping assignment I' based on a proposal distribution $Q(G', I' | G^{l-1}, I^l)$. We describe the design of this proposal distribution in Appendix A.2. We then finetune the reward function conditioned on the proposal to obtain new parameters θ', which defines a new reward function $R(s|G';\theta')$. To do that, we use two types of optimization. First is reward regression based on the state and reward pairs $(s, r) \in S_D$. Since we obtained equivalence states from the proposed mappings, the reward function can be optimized so that for each state s in S_D, all of its equivalent states will have the same reward as s itself. We formally define a regression loss as follows

$$\mathcal{L}(\theta)_{\text{reg}} = \mathbb{E}_{(s,r)\sim S_D}[(R(\Phi(s, I')|G';\theta) - r)^2]. \quad (6)$$

The second type of optimization is reward ranking. Specifically, we optimize the reward function so that the reward of a state $s_+ \in S_+$ is higher than the reward of any state $s_- \in S_-$. This gives us the second loss:

$$\mathcal{L}(\theta)_{\text{rank}} = \mathbb{E}_{s_+\sim S_+, s_-\sim S_-}[|(R(\Phi(s_-, I')|G';\theta) - R(\Phi(s_+, I')|G';\theta)|_+].$$

We then combine these two loss functions to update the parameters of the reward function:

$$\theta' = \arg\min_\theta \mathcal{L}(\theta)_{\text{reg}} + \mathcal{L}(\theta)_{\text{rank}}. \quad (7)$$

Based on the new reward, we generate a query to reflect the change in the hypothesis and in the corresponding reward function. The query is a goal state s_q' sampled starting from the current state (i.e., s_q^{l-1}). Intuitively, this new state should have a high reward based on the new reward function but a low reward based on the previous reward function at iteration $l-1$. Formally, s_q' is sampled by

$$s_q' = \arg\max_{s\in\mathcal{N}(s_q^{l-1})} R(s|G';\theta') - \lambda R(s|G^{l-1};\theta^{l-1}), \quad (8)$$

where $\mathcal{N}(s_q^{l-1})$ is the set of all reachable states starting from s_q^{l-1} and λ is a constant coefficient.

After reaching the new query state s_q', we query for oracle feedback by asking if s_q^l is an acceptable state that satisfies the goal. If acceptable, we then accept the new proposal as well as the new reward function, and augment S_+ with the new query s_q'. We also assume that s_q' has a similar reward as the final demonstration state. Thus S_D can also be augmented with (s_q', r^T), where r^T is the reward of the final demonstration state according to the initial reward function. If the oracle feedback is negative, then we reject the new proposal and the corresponding reward function, move back to the last accepted state (i.e., s_q^{l-1}), and augment S_- with s_q'. We repeat this process until reaching the maximum number of iterations L_{max}. Alternatively, we can also terminate the process when no sparser graphs have been accepted for a certain amount of iterations.

After the two learning phases, we apply the learned reward function to the test environments by sampling a goal state based on the reward function. Since we prefer sparse graphs, we use the most sparsest graphs accepted in the recent iterations. When there are multiple accepted graphs with the same number of edges, we use the one that has the most mappings assigned to its edges.

5. Experiments

5.1. Watch&Move Environment

We propose a one-shot imitation learning environment, Watch&Move. Inspired by recent machine learning benchmarks based on 2D physics engines (Lowe et al., 2017;

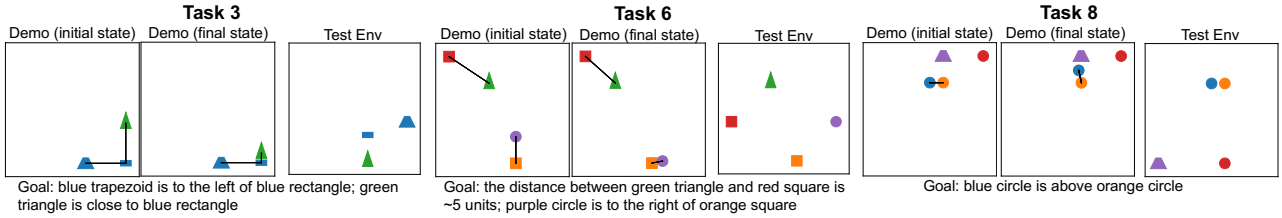

Figure 4. Example Watch&Move tasks. For each task, we create an expert demonstration that moves objects to a state that satisfies the goal and provide a test environment that is different from the initial state in the expert demonstration. The edges visualize the relationships that are part of the goal definition; they are not present in the expert demonstration and are thus unknown to the agent. We also show the goal for each task. The distance in Watch&Move is measured in units. As a reference, the radius of the circle is 0.8 units. The exact definitions of the goal relationships are summarized in Table 1 (Appendix A.6).

Bakhtin et al., 2019; Allen et al., 2020; Netanyahu et al., 2021), we create a 2D physics simulation for Watch&Move, where objects can be moved w.r.t. physics dynamics and constraints. This simulation environment provides an abstraction of real world object rearrangement tasks[2].

We design 9 object rearrangement tasks in the Watch&Move environment (Figure 4 shows 3 example tasks). Each task consists of an expert demonstration and a new testing environment. The goals of the proposed tasks cover a range of objects relationships that are common in the real world as well as in the prior work on object rearrangement tasks (as summarized in Table 1).

A successful performance in the testing environment in Watch&Move requires an agent to rearrange the objects to satisfy all spatial goal relationships while minimizing the overall change in the environment. This requires that the agent correctly identify the necessary relationships for each task. We use a reward function to measure the task completion and the displacement of all objects: $R_{\text{eval}} = \mathbb{1}(s^T \text{ satisfies the goal}) - 0.02 \sum_{i \in N} ||x_i^0 - x_i^T||_2$, where x_i^0 and x_i^T are the first and last states of object i in a testing episode respectively, and s^T is the overall final state in the testing episode. Please refer to Appendix A.6 for more details about the tasks and the environment.

5.2. Baselines and Ablations

We compare GEM with M-AIRL (i.e., the initial reward learning alone) and with ReQueST (Reddy et al., 2020), a recent active reward learning approach that estimates a reward that can generalize to environments with different initial state distributions. For a fair comparison, we use the same graph-based reward function for ReQueST and initialize the reward for using the expert demonstration. For details, please refer to Appendix A.7. For the ablated study, we also evaluate variants of GEM, including i) GEM without minimizing the previous reward for the query generation, ii) GEM with a fully connected graph (no new graph proposals), iii) GEM without applying any equivalence mappings, and iv) GEM trained with randomly generated queries. During testing, given the reward function learned by each method, we sample a goal state by maximizing the reward and minimizing the displacement. We then evaluate sampled goal states based using our reward metric defined in Section 5.1. Finally, we provide an oracle performance based on the optimal goal states generated by the oracle in the testing environments.

5.3. Results with a Simulated Oracle

We first conducted an evaluation on 8 Watch&Move tasks with a simulated oracle that gives feedback for a query based on whether the state in the query satisfies the goal definition. We report the reward obtained in the testing environment using models trained with different methods with different numbers of queries in Figure 5. The results show that the initial reward trained by M-AIRL failed to reach the goal in the tesing environment. However, with the active reward refinement enabled by GEM, the reward function was greatly improved. We also found that the graphs and the equivalence mapping assignments inferred by GEM provided meaningful representations of the intended spatial relationships for the tasks (we visualize the inferred graphs and equivalence mapping assignment, and example queries in Appendix B). In comparison, ReQueST and the random query variant failed to learn a generalizable reward function for all tasks. Other ablated variants could achieve success in some tasks but not in all 8 tasks, which verified the importance of i) minimizing the previous reward for the query sampling and ii) the joint inference of the graph structure and the equivalence mapping assignment.

[2]We illustrate this in Appendix B.1.

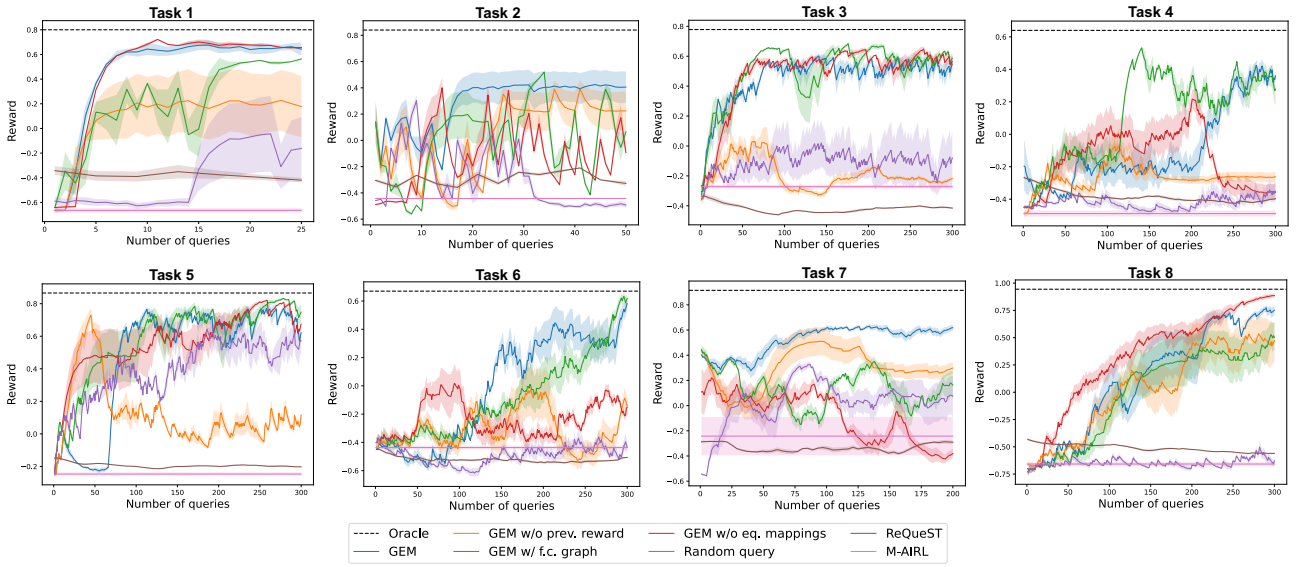

Figure 5. Testing performance of different methods trained with a simulated oracle on 8 Watch&Move tasks (Task 1 to 8). We plot the reward metric in the testing environment using the learned model as a function of the number of queries. The dashed line indicates the reward for an optimal plan generated by the oracle. Note here the M-AIRL baseline provides the initial rewards for GEM and all of its variants and is not updated with the queries. We ran each method using three random seeds and show the standard errors as the shaded areas.

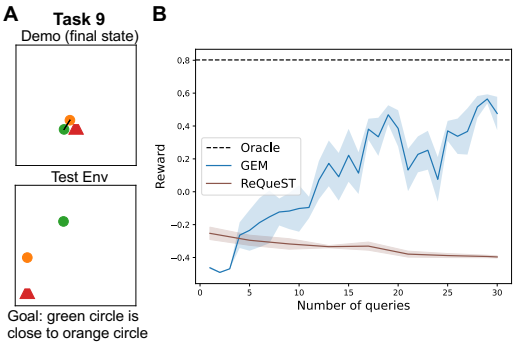

Figure 6. (**A**) Illustration of Task 9 used in the human experiment. (**B**) Testing performance of GEM and ReQueST trained with three human oracles for Task 9. The dashed line indicates the reward for an optimal plan generated by the oracle in the test environment. The shaded areas indicate the standard errors.

5.4. Human Experiment

To evaluate how well GEM can work with real human oracles, we conducted a human experiment on Task 9, where we recruited three human participants as oracles. The participants gave their consent and the study was approved by an institutional review board.

At each trial, a participant was instructed to verify whether a query state generated by an AI agent satisfied the ground-truth goal of Task 9 (see Figure 6). Each participant interacted with GEM and ReQueST once, and provided feedback for 30 queries for each algorithm. Each GEM query took about 1 minute due to training. We plot the rewards in the testing environment in Figure 6, which demonstrates that the reward function trained by GEM reaches a good performance in the new environment within 30 iterations, significantly outperforming the reward function trained by ReQueST. GEM also correctly inferred the necessary edge (the pair of circles) and its corresponding invariance type (rotation-invariant).

6. Conclusion

We have proposed a reward learning algorithm, GEM, for performing one-shot generalization for an object rearrangement task. In GEM, spatial goal relationships are represented by a graph-based reward function and state equivalence mappings assigned to the edges in the graph. GEM infers the graph structure and the equivalence mapping assignment by combing an initial reward learning using inverse reinforcement learning and a sample efficient active reward refinement. For evaluation, we designed a 2D physics simulation environment, Watch&Move, and compared GEM against strong baselines on multiple tasks. We also conducted a human experiment to verify whether GEM can work with humans. The experimental results show that GEM was able to discover meaningful spatial goal represen-

tations. It significantly outperformed baselines, achieving a better generalization in unseen testing environments as well as a greater sample efficiency. We believe that GEM is a step towards solving the extremely challenging problem of one-shot imitation learning.

The success of GEM in simulation shows potential for real world applications. First, GEM achieves a much greater sample efficiency and generalizability compared with SOTA (ReQueST), making it possible to learn the reward with humans. Second, there are several ways to apply GEM to learn task specifications that involve more objects with a reasonable amount of queries. The bottleneck of the sample efficiency is the number of edges in a graph. Since real world tasks typically require a sparse graph, we can use heuristics (e.g., an object is usually only related to neighboring objects) to directly remove unlikely edges without queries. We can also take a hierarchical approach where we first learn rewards for subgraphs (e.g., a dinning set) and then apply GEM to learn the final reward (e.g., multiple sets for a party of four) where each node is a subgraph. Finally, since our reward representation and the learning algorithm only require generic object states (e.g., positions, orientations), it is possible to learn rewards in different state spaces for different domains (such as shapes in a 3D space rather than shapes in a 2D space). We intend to study the real world applications of GEM in the future.

The present work has a few limitations. First, we are only focusing on pairwise relationships between objects and representing them through two types of invariances. In the future, we intend to learn higher-order relationships by enabling message passing in the graphs and introduce additional types of invariance. Second, for real-world applications, we need to improve proposal sampling so that it can infer the graph structure and the equivalence mapping assignment more efficiently for a large number of objects. We can potentially achieve this by using language instructions to guide the inference. For instance, a human user can provide language description of the task in addition to the physical demonstration to help achieve a better understanding of the task.

Acknowledgements

We would like to thank Xavier Puig for his assistance on the experiment in VirtualHome. This work was supported by the DARPA Machine Common Sense program and ONR MURI N00014-13-1-0333.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Allen, K. R., Smith, K. A., and Tenenbaum, J. B. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47):29302–29310, 2020.

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

Arumugam, D., Lee, J. K., Saskin, S., and Littman, M. L. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257*, 2019.

Baker, C. L., Jara-Ettinger, J., Saxe, R., and Tenenbaum, J. B. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1(4):1–10, 2017.

Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., and Girshick, R. Phyre: A new benchmark for physical reasoning. *arXiv preprint arXiv:1908.05656*, 2019.

Billard, A., Epars, Y., Calinon, S., Schaal, S., and Cheng, G. Discovering optimal imitation strategies. *Robotics and autonomous systems*, 47(2-3):69–77, 2004.

Bıyık, E., Palan, M., Landolfi, N. C., Losey, D. P., and Sadigh, D. Asking easy questions: A user-friendly approach to active reward learning. *arXiv preprint arXiv:1910.04365*, 2019.

Bonardi, A., James, S., and Davison, A. J. Learning one-shot imitation from humans without humans. *IEEE Robotics and Automation Letters*, 5(2):3533–3539, 2020.

Brown, D., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pp. 783–792. PMLR, 2019.

Cao, Z., Gao, H., Mangalam, K., Cai, Q.-Z., Vo, M., and Malik, J. Long-term human motion prediction with scene context. In *European Conference on Computer Vision*, pp. 387–404. Springer, 2020.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Cui, Y. and Niekum, S. Active reward learning from critiques. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6907–6914. IEEE, 2018.

Daniel, C., Viering, M., Metz, J., Kroemer, O., and Peters, J. Active reward learning. In *Robotics: Science and systems*, volume 98, 2014.

Daniel, C., Kroemer, O., Viering, M., Metz, J., and Peters, J. Active reward learning with a novel acquisition function. *Autonomous Robots*, 39(3):389–405, 2015.

Duan, Y., Andrychowicz, M., Stadie, B., Jonathan Ho, O., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *Advances in neural information processing systems*, 2017.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pp. 357–368. PMLR, 2017.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Ghasemipour, S. K. S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pp. 1259–1277. PMLR, 2020.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, 2013.

Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pp. 4399–4410. PMLR, 2020.

Huang, D.-A., Xu, D., Zhu, Y., Garg, A., Savarese, S., Fei-Fei, L., and Niebles, J. C. Continuous relaxation of symbolic planner for one-shot imitation learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2635–2642. IEEE, 2019.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Loftin, R., MacGlashan, J., Peng, B., Taylor, M., Littman, M., Huang, J., and Roberts, D. A strategy-aware technique for learning behaviors from discrete human feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Wang, G., Roberts, D. L., Taylor, M. E., and Littman, M. L. Interactive learning from policy-dependent human feedback. In *International Conference on Machine Learning*, pp. 2285–2294. PMLR, 2017.

Nehaniv, C. L., Dautenhahn, K., et al. The correspondence problem. *Imitation in animals and artifacts*, 41, 2002.

Netanyahu, A., Shu, T., Katz, B., Barbu, A., and Tenenbaum, J. B. Phase: Physically-grounded abstract social events for machine social perception. In *35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

Puig, X., Shu, T., Li, S., Wang, Z., Liao, Y.-H., Tenenbaum, J. B., Fidler, S., and Torralba, A. Watch-and-help: A challenge for social perception and human-{ai} collaboration. In *International Conference on Learning Representations*, 2021.

Reddy, S., Dragan, A., Levine, S., Legg, S., and Leike, J. Learning human objectives by evaluating hypothetical behavior. In *International Conference on Machine Learning*, pp. 8020–8029. PMLR, 2020.

Ross, S., Gordon, G., and Bagnell, D. 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, 2011.

Rowe, R., Singhal, S., Yi, D., Bhattacharjee, T., and Srinivasa, S. S. Desk organization: Effect of multimodal inputs on spatial relational learning. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1–8. IEEE, 2019.

Schaal, S. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

Shah, A., Kamath, P., Shah, J. A., and Li, S. Bayesian inference of temporal task specifications from demonstrations. In Bengio, S., Wallach, H., Larochelle, H.,

Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3804–3813. Curran Associates, Inc., 2018.

Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., Liu, K., et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pp. 477–490. PMLR, 2022.

Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*, 2016.

Sun, J., Yu, L., Dong, P., Lu, B., and Zhou, B. Adversarial inverse reinforcement learning with self-attention dynamics model. *IEEE Robotics and Automation Letters*, 6(2): 1880–1886, 2021.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Wang, S., Toyer, S., Gleave, A., and Emmons, S. The `imitation` library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation, 2020.

Wang, X., Lee, K., Hakhamaneshi, K., Abbeel, P., and Laskin, M. Skill preferences: Learning to extract and execute robotic skills from human feedback. In *Conference on Robot Learning*, pp. 1259–1268. PMLR, 2022.

Xu, K., Ratner, E., Dragan, A., Levine, S., and Finn, C. Learning a prior over intent via meta-inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 6952–6962. PMLR, 2019.

Yan, F., Wang, D., and He, H. Robotic understanding of spatial relationships using neural-logic learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8358–8365. IEEE, 2020.

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

A. Implementation Details

A.1. State Equivalence Mappings

Applying a mapping to an edge once. When applying a mapping to an edge, we randomly sample a variable necessary for the mapping. For the the rotation-invariant mapping, $\phi_1(x_1, x_2)$, we sample a random angle $d \sim \text{Uniform}(-\pi, \pi)$, and rotate i around j by the angle of d. For the scale-invariant mapping ϕ_2, we randomly sample a scale $\rho \sim \text{Uniform}(0.1, 10.0)$, and move i so that its distance from j changes by the scale of ρ while the relative orientation from i to j remains the same.

State transformation given all mappings. For each graph, there is a corresponding mapping assignment for all edges $I = \{\delta_{ijk}\})_{(i,j)\in E, k\in K}$. We apply a mapping ϕ_k to edge (i, j) when and only when $\delta_{ijk} = 1$. If multiple mappings are assigned to an edge, they will be applied recursively. E.g., if ϕ_1 and ϕ_2 are assigned to (i, j), the final state transformation for this edge is $\phi_2(\phi_1(x_i, x_j))$. Let $\tilde{x}_{ij}$ be the transformed edge, then the final transformed state is $\Phi(s, I) = \{\tilde{x}_{ij}\}_{(i,j)\in E}$, and the corresponding reward becomes:

$$R(\Phi(s, I)|G) = \frac{1}{|E|} \sum_{(i,j)\in E} r(\tilde{x}_{ij}). \tag{9}$$

Note that since the state transformations are applied to each edge independently, any change in one edge will not affect other edges. During reward finetuning, we apply state transformation based on the proposed mapping assignment to each batch, so that the trained reward function reflects the intended invariance represented in the assigned equivalence mappings for each edge.

Additionally, we assume that the absolute coordinates of the objects do not matter in the goal specifications. Therefore, we also apply a random shift to all objects' coordinates after applying the state equivalence mappings (i.e., moving all objects together without changing their relative positions).

A.2. Proposal Sampling

Each proposal consists of a new graph and a new equivalence mapping assignment. Therefore, there are two general types of proposal sampling – (1) graph sampling and (2) equivalence mapping assignment sampling. We use a prior probability q_{type} to decide the type of sampling for each iteration. At each iteration, we first sample $u \sim \text{Uniform}(0, 1)$. If $u < q_{\text{type}}$, we choose to sample a new equivalence mapping assignment; otherwise, we sample a new graph. For 3-object tasks, we use $q_{\text{type}} = 0.2$; for 4-object tasks, we use $q_{\text{type}} = 0.5$.

To sample a new graph, we can either add an edge or remove an edge. We define the chance of removing an edge as q_{remove}. Then we sample $u \sim \text{Uniform}(0, 1)$. When $u < q_{\text{remove}}$, we sample one of the edges from G^{l-1} to remove; otherwise, we randomly add an edge that does not exist in G^{l-1}. Note that we consider undirected graphs in this work; we also avoid removing all edges to ensure a valid graph-based reward function. For all tasks, we use $q_{\text{remove}} = 0.5$.

To sample a new equivalence mapping assignment, we uniformly sample an edge $(i, j) \in E$ and a type of mapping $k \in K$, and change the corresponding assignment, i.e., $\delta'_{ijk} = 1 - \delta^{l-1}_{ijk}$.

A.3. Network Architecture

The discriminator reward and value networks are implemented by a graph-based architecture, as opposed to the MLP architecture used in the original AIRL version. The input is the observation representation s and graph G. We construct edge representations by concatenating every pair of object representations in the observation. We then apply 4 x (fully connected layers + ReLU) to each edge representation. We apply a final fully connected layer to each edge embedding to output a single value for each edge. The final reward is the average edge value *only* for edges in G.

A.4. Training Details and Hyperparameters

Model-based state-based AIRL: We build upon an AIRL implementation (Wang et al., 2020) and add a model-based generator. For each task, M-AIRL is executed for 500k generator steps, the expert batch size is the length of the expert demonstrations. For the model-based policy, we set $\beta = 0.3$ in Eq. (4). The discriminator is updated for 4 steps after every model-based generator execution. The model based generator samples approximately 2k steps on each iteration.

Query reward finetuning: We apply 5k network updates per query iteration. For optimizing the network, we use Adam

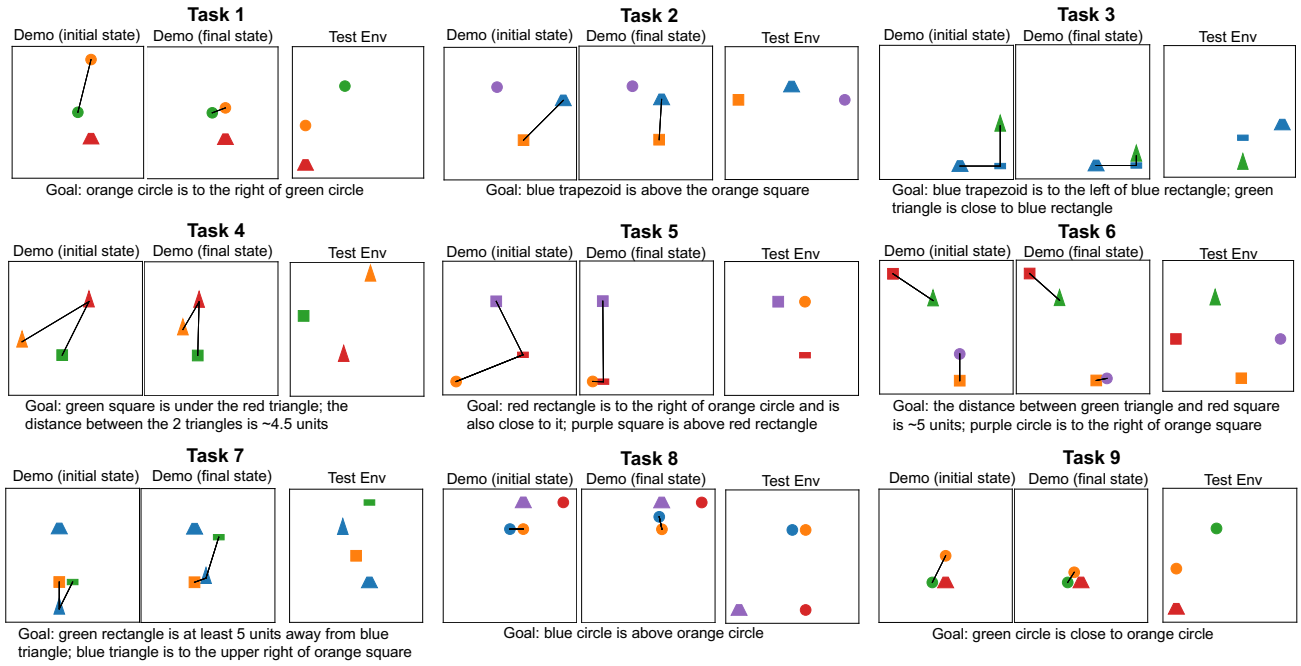

Figure 7. Illustration of all 9 Watch&Move tasks. For each task, we show the demonstration, the testing environment, and the goal definition. We also show the goal relationships by visualizing the corresponding edges.

optimizer (Kingma & Ba, 2014) with a learning rate of 0.0003. For each update, we sample a batch of 16 states for the regression loss and a batch of 16 pairs of positive and negative states for the reward ranking loss.

Query selection: we set $\lambda = 0.2$ in Eq 8.

Random query variant: we always assume a fully connected graph and do not assign equivalence mappings to the edges. We collect the positive and negative sets and refine the reward function using the same loss function defined in Eq. (7).

A.5. Sampling Goal States in Testing

During testing, we sample a goal state, $s^* = (x_i^*)_{i \in N}$, by jointly maximizing the learned reward and minimizing the displacement. I.e.,

$$s^* = \underset{s=(x_i)_{i \in N}}{\arg\max}\ R(s, G) - 0.02 \sum_{i \in N} ||x_i^0 - x_i||_2. \tag{10}$$

For the experiments using the simulated oracle, we use the reward corresponding to the sparsest graph accepted that has been accepted up until the current iteration. For the human experiments, we directly use the reward accepted at each iteration so that the results are more resilient to the noise in participants' responses.

Since this work focuses on learning goals, we directly evaluate feasible sampled goal states in the experiments. However, it is also possible to evaluate an episode using a planner to reach the sampled goal states.

A.6. Watch&Move Environment

Figure 7 illustrates the setup (the expert demonstration, the test environment, and the goal description) for each Watch&Move task. The demonstrations of these tasks present various sources of confusion. For instance, there can be irrelevant objects (e.g.,the purple trapezoid and the red circle in Task 8) or goal objects that are never moved in the demonstration due to their initial states being satisfactory (e.g., the blue trapezoid and the blue rectangle in Task 3). Expert demonstrations were

Table 1. Definition of goal relationships in Watch&Move tasks. For reference, the radius of a circle is 0.8 units.

PAIRWISE GOAL RELATIONSHIP	DEFINITION
CLOSE [(SRIVASTAVA ET AL., 2022)]	OBJECTS DISTANCE < 2.5 UNITS
LEFT/RIGHT [(JOHNSON ET AL., 2017; YAN ET AL., 2020)]	ANGLE BETWEEN OBJECTS AND X AXIS $< 0.1\pi$
ABOVE/BELOW [(YAN ET AL., 2020)]	$0.4\pi <$ ANGLE BETWEEN OBJECTS AND Y AXIS $< 0.6\pi$
DIAGONAL	BOTH COORDINATES OF ONE OBJECT $>$ OTHER'S
DISTANCE X	OBJECTS DISTANCE WITHIN 0.5 UNIT BUFFER AROUND X
AT LEAST WITHIN DISTANCE X	OBJECTS DISTANCE $>$ X

created with a planner introduced in (Netanyahu et al., 2021), with a length ranging from 8 to 35 steps.

The state space in Watch&Move is represented by $s \in \mathbb{R}^{N\times 13}$ where N is the number of objects in the environment. 13 dimensions are composed of the coordinate of the object center, the object's angle, and one hot encodings of the object shape and color. The action space is discrete, containing 11 possible actions per object (8 directions, turning right and left and stopping), and the object id. The action space is proportional to the number of objects. We use PyBox2D[3] to simulate the physical dynamics in the environment.

The goal relationships used to create Watch&Move are specified in Table 1. These could be easily extended to any pairwise spatial relation, such as touching, covering, no contact, orientation, etc.

A.7. ReQueST

ReQueST (Reddy et al., 2020) is a method for estimating a reward ensemble that can safely generalize to environments with different initial state distributions. ReQueST generates queries from a generative model that optimizes four objectives. Each state in each query receives accepted or rejected feedback from an oracle and is used as a positive or negative example in reward training. To ensure a fair comparison with our approach, we implement ReQueST with the following changes.

- The original method does not use an expert demonstration, therefore we pre-train the ensemble reward functions with the expert demonstration.
- The reward architecture is similar to ours, where the final edge embeddings are sum-pooled and fed to a fully connected layer followed by softmax for the classification. Note that here we only have two classes – neutral and good.
- The original paper learns a world model from random sampling in the environment. We use the ground truth world model provided by the physics simulation, similar to GEM.
- Each query is of length 1 (as in the pointmass environment in ReQueST) sampled similarly to the AIRL generator sampling in GEM. Starting from the final state of the expert demonstrations, each query is sampled relative to the last sampled point, matching with the query generation procedure in GEM.

We use an ensemble of 4 rewards (as in the pointmass environment in ReQueST), and train them using the Adam optimizer with a learning rate of 0.0003. We have 1k pre-training steps and 10 training steps per every 4 queries.

B. Additional Results

B.1. Evaluation in a Real World Task

To show the connection between Watch&Move tasks and real world tasks, we use VirtualHome (Puig et al., 2018; 2021), a realistic virtual household simulation platform, to instantiate Task 5 in a real work setting – setting up a desk for a left-handed person, i.e., a book in the front of a notepad and a pen on the left of and close to the notepad. In Figure 8, we show the reward trained by GEM on the original Task 5 can also achieve success in this real-world task which shares the same ground-truth goal with Task 5 but requires the rearrangement of real world objects.

[3]https://github.com/pybox2d/pybox2d

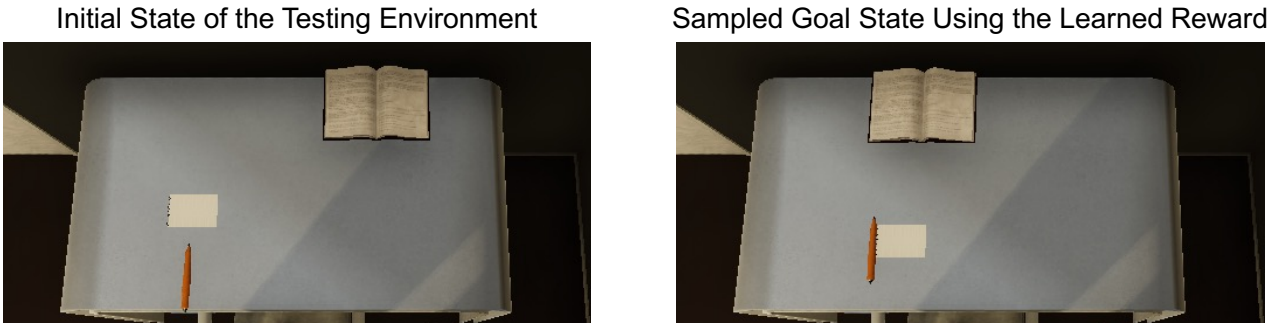

Figure 8. A real world task simulated in VirtualHome that has the same goal specification as Task 5. Specifically, the task is to set up a desk for a left-handed person to take notes while reading a book. We visualize the testing environment and the sampled goal state for this real world task using the reward function learned by GEM for Task 5. The sampled goal state satisfies the true goal specification while minimizing the displacement of the objects.

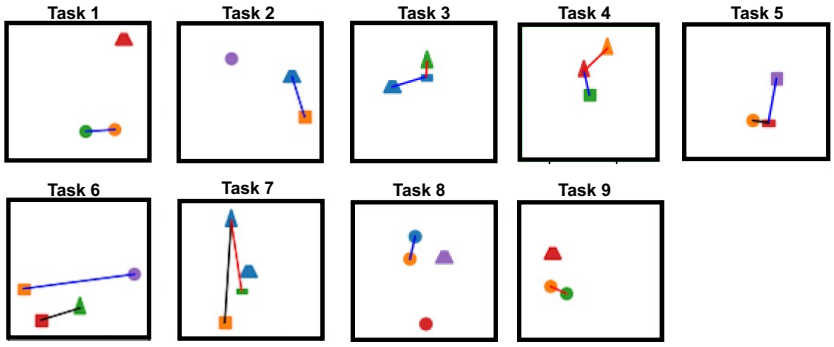

Figure 9. Illustration of the inferred graphs and equivalence mapping assignments as well as the corresponding query states that lead the proposal acceptance for all 9 tasks (from one of the three runs). The colors of the edges indicate the assigned mappings. Black: no mapping is assigned; red: the rotation-invariant mapping is assigned; blue: the scale-invariant mapping is assigned; purple: both the rotation-invariant and the scale invariant mappings are assigned.

B.2. Learning Results

We show the inferred graphs and the equivalence mapping assignments for all 9 tasks in Figure 9. The inferred graphs correctly identified the goal relationships. The assigned equivalence mappings also revealed the invariance properties of the intended spatial relationships in most cases.

Figure 10 depicts typical queries that are sampled by GEM to verify the proposed graphs and the equivalence mapping assignment. The examples here demonstrate how the informative queries were able to help differentiate two different reward functions.

B.3. Qualitative Results

Figure 11 shows the sampled goal states using the reward functions learned by GEM. These goal states not only satisfy the goal specifications, but are also efficient (small displacement compared to the initial states in the testing environments). This demonstrates that the sampled goal states are not simply copying the final state of the expert demonstration but indeed reflect the intended spatial relationships.

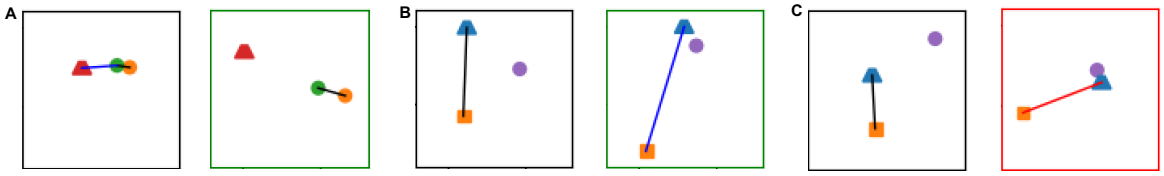

Figure 10. (**A**)-(**C**) are example queries showing the effect of the new graph and equivalence mapping assignment. Each example first shows the state and proposal before the query, and then shows the new proposal and the sampled query state. The colors of the edges indicate the assigned mappings. Black: no mapping is assigned; red: the rotation-invariant mapping is assigned; blue: the scale-invariant mapping is assigned; purple: both the rotation-invariant and the scale-invariant mappings are assigned. The colors of the boxes indicate oracle feedback (red: reject, green: accept). In (**A**), an edge was removed, and the irrelevant object was consequently placed far away from the remaining objects. The example in (**B**) shows that when the scale-invariant mapping was assigned to an edge, the sampled query changed the distance between the objects connected by that edge while generally preserving the relative orientation between the two. On the other hand, when the rotation-invariant mapping was assigned as in (**C**), one of the objects was rotated around the other connected object and there was little change in the distance between the two objects.

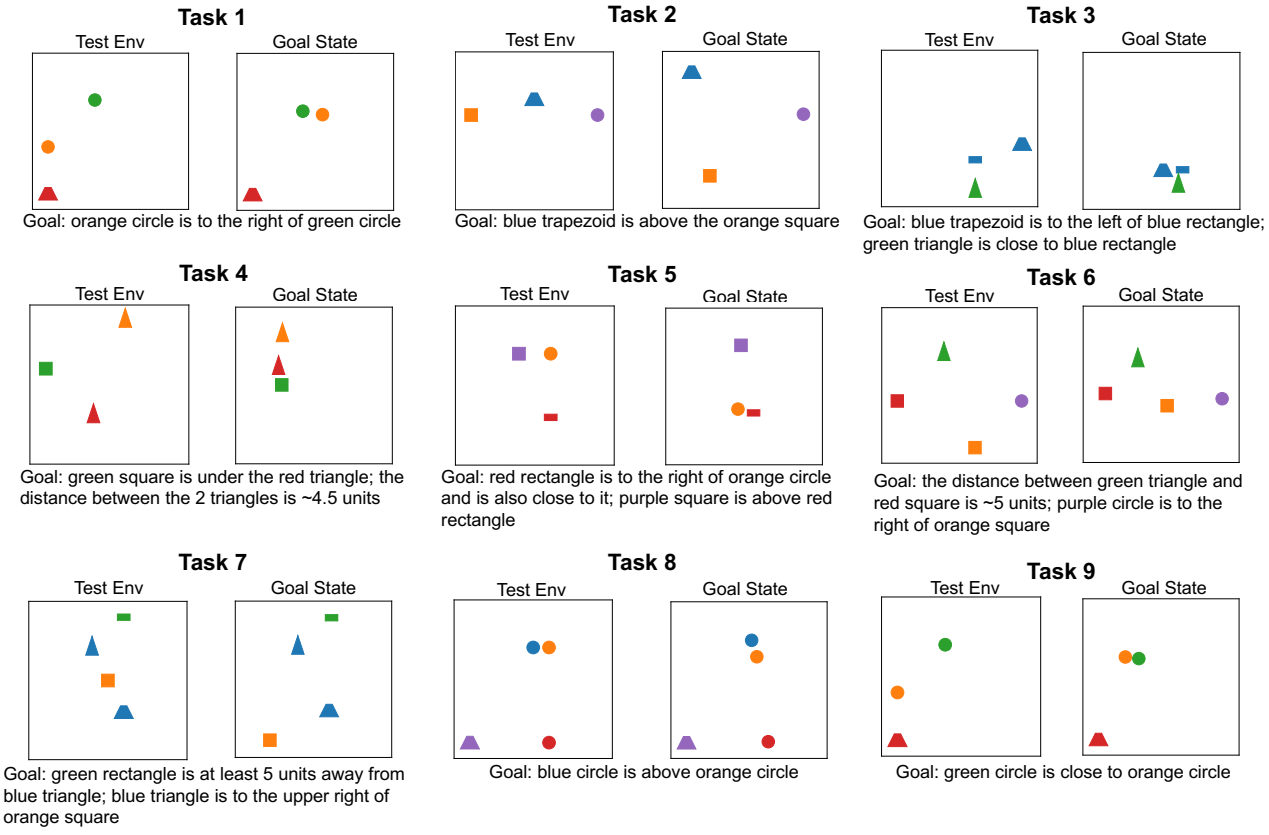

Figure 11. The testing environments and the sampled goal states based on the reward functions learned by GEM for the corresponding tasks.