DO AS THE ROMANS DO: LEARNING UNIVERSAL BEHAVIORS FROM HETEROGENEOUS AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans often acquire new skills by observing others, since observed behaviors implicitly reveal how to act reasonably in an environment. However, observations drawn from a heterogeneous population introduce conflicting behavioral signals, making it difficult to determine which behaviors are worth imitating. We address this challenge with General Reward Inference and Disentanglement (GRID), a social learning method that extracts universally useful behaviors from a heterogeneous population of demonstrators pursuing different goals. GRID decomposes per-agent reward functions into a general reward, capturing behaviors shared across all agents, and specific rewards, capturing individual preferences and objectives, through an information bottleneck. Training exclusively on the general reward provides a new paradigm of generalist pretraining. It yields a generalist agent that internalizes universal environmental competencies, such as safety and basic task proficiency, without the mode-averaging bias that afflicts standard learning from demonstration techniques. This generalist serves as a strong prior for fine-tuning to downstream tasks, including preferences unseen during training. Experiments across a synthetic basis function decomposition, multi-agent Craftax, continuous control tasks (MuJoCo Gym) and an autonomous driving simulator (Highway-Env) confirm that GRID successfully disentangles reward structure in a semantically meaningful way, outperforms standard learning from demonstration baselines, and enables more efficient and stable specialization.

1 INTRODUCTION

"When in Rome, do as the Romans do[1]"—but *which* Roman should we follow? Observed behaviors of individuals implicitly reveal information about how one should act in the environment, yet not all agents follow the same objective, and blindly imitating any one of them can be dangerous or counterproductive. Consider a robot placed among a crowd at a traffic intersection: different pedestrians have different destinations, so they cross the street in different directions—but everyone respects the crosswalk signal. The robot must infer that the crosswalk status is the relevant shared cue, while individual destinations and the behaviors that pursue them are not. Or consider a robot navigating a building where a patch of floor along the shortest path is slippery. Humans will naturally avoid this patch, taking a longer route regardless of their destination. An intelligent agent does not need to know any individual's goal to extract something valuable: the avoidance behavior is universal across all agents. By identifying what every agent consistently does, independent of their individual objectives, a learning agent can infer what is broadly important and adopt those behaviors itself.

The challenge is extracting these shared signals from a heterogeneous population of demonstrators. Standard learning from demonstration (LfD) is vulnerable to potentially conflicting, multi-modal demonstrations and demands additional mechanisms to address variations (Seraj et al., 2024). Inverse reinforcement learning (IRL), a subset of LfD, recovers an unknown reward function. However, when a population pursues heterogeneous objectives, standard IRL algorithms tend to average conflicting goals together, producing a suboptimal policy that fails to achieve any specific task effectively (Poddar et al., 2024). In contrast, reinforcement learning (RL) learns directly from environmental feedback through a known reward function (Sutton & Barto, 2018). However, RL is typically sample-inefficient and brittle to environmental changes, leading to poor generalization. In

[1]The proverb advises us to follow the customs and traditions of the place we are visiting.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

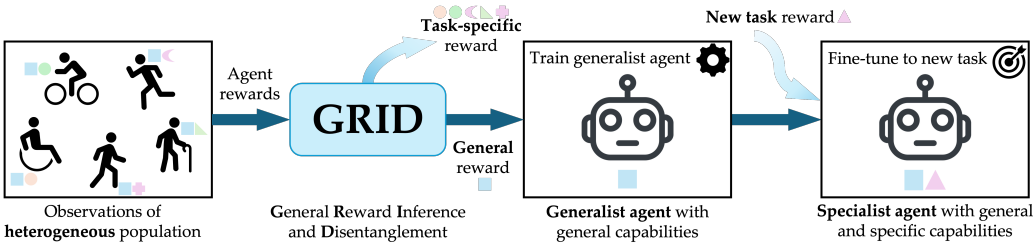


Figure 1: General Reward Inference and Disentanglement (GRID) is a social learning method that extracts universally useful behaviors from a heterogeneous population of demonstrators pursuing different goals. The learned general behavior can be used to train a generalist agent, which can then be fine-tuned for a new task.

short, neither RL nor standard LfD gracefully handles learning from a population of agents pursuing different goals, even though these agents may reveal extremely valuable information.

This raises a fundamental question: *What should an agent learn from such a population*? We posit that the most valuable signals are those *shared across agents*—behaviors that are universal precisely because they reflect general competence, safety, and environmental norms, rather than any individual's specific objective. All pedestrians obey the crosswalk signal regardless of destination. All skilled drivers maintain safe following distances regardless of where they are headed. All navigators avoid the slippery floor regardless of their target. These shared behaviors are the "Roman customs" worth adopting—general norms of the environment that any capable agent implicitly follows.

Social learning, a combination of RL and IRL, moves in this direction. An agent observes other agents in the environment to infer relevant environment information and objectives, and uses that knowledge to inform its own behavior through its own interactive experiences with the environment. Prior works (Filos et al., 2021; Abdulhai et al., 2023) explicitly disentangle shared features of the environment from agent-specific goals, but do not isolate a shared reward signal that represents universally desirable behaviors. To our knowledge, no existing work explicitly disentangles the underlying reward of heterogeneous agents into universally shared competencies and individual preferences. We address this gap with **General Reward Inference and Disentanglement (GRID)**.

GRID features a novel deep learning architecture that uses an information-theoretic objective to simultaneously capture individual-specific behaviors and the general behaviors common across agents. Concretely, we assume individual reward functions, available via RL environments or IRL, express individual behaviors. Then, we decompose each into a sum of a general reward and an individual-specific reward. The general reward captures a universally shared structure: what every competent agent does regardless of their personal goal. Performing RL with this general reward yields a generalist agent that has internalized the environmental norms of the observed population. Crucially, this agent serves as a strong prior for downstream adaptation as it already knows what to do and what to avoid and can be fine-tuned for any specific task more efficiently than training from scratch. Unlike traditional specialist agents that must learn fundamental physics and safety constraints from scratch for each new objective, our generalist agent already knows how to operate safely in the environment and can be efficiently fine-tuned to specific tasks.

Statement of Contributions. **(1)** We introduce the GRID architecture and information-theoretic training objective that jointly disentangles general and specific reward structures from a heterogeneous population of demonstrators pursuing different goals. **(2)** We provide intuition on general and specific reward structures through interpretable visualizations of the learned reward decomposition across synthetic, multi-agent, and robotic domains, confirming semantic alignment with ground-truth reward components. **(3)** We demonstrate that the GRID generalist outperforms LfD baselines in broadly applicable competencies and downstream fine-tuning to unseen tasks.

2 Related Works

Behavior learning. Standard LfD methods (Torabi et al., 2018; Ross et al., 2011; Abbeel & Ng, 2004; Ziebart et al., 2008; Ramachandran & Amir, 2007) and powerful generative approaches (Ho &

Ermon, 2016; Fu et al., 2018) share a common assumption that all demonstrations originate from a single, consistent demonstrator. However, in large datasets and in the real world, demonstrations are often heterogeneous, reflecting different skills, preferences, and goals across demonstrators. Later works address heterogeneity through mutual information maximization (Li et al., 2017; Yu et al., 2019; Schrum et al., 2021; 2024; Poddar et al., 2024), while others use CVAEs and variational inference for multi-modal human behavior modeling, common in autonomous driving applications (Schmerling et al., 2018; Ivanovic et al., 2021; Remy et al., 2026). However, all of these approaches treat heterogeneity as a modeling target—the goal is to explain variations between demonstrators. In contrast, our approach explicitly learns the shared commonalities between demonstrators.

Reward decomposition. GRID contributes to a growing interest in structured reward decomposition. Multi-Style Reward Distillation (MSRD) is the closest in spirit to our work (Chen et al., 2020a). MSRD is an IRL algorithm that learns to perform a single task from heterogeneous demonstrations by decomposing the total reward into a shared task component and an individual strategy component via neural network distillation. Unlike the aforementioned LfD methods, MSRD can explain both the heterogeneity and shared commonalities of various demonstrations. GRID also separates a shared reward from individual rewards, but does not assume all demonstrations are of the same task. In fact, GRID explicitly handles heterogeneous tasks by performing reward decomposition after the IRL procedure instead of during as MSRD does. Furthermore, GRID's approach is grounded in information theory whereas MSRD is not. Fast Lifelong Adaptive Inverse Reinforcement learning (FLAIR) and Reward Decomposition From Suboptimal and Heterogeneous Demonstrations (REPRESENT) build off of MSRD to better handle various strategies over time (Chen et al., 2022) and suboptimal demonstrations (Xue et al., 2025) respectively. Other works feature hand-engineered reward decomposition (van Seijen et al., 2017), redistribute rewards temporally along relevant points in the episode (Arjona-Medina et al., 2019), or seek interpretable decompositions of rewards (Juozapaitis et al., 2019). GRID unifies these threads: the decomposition is learned without manual reward engineering, is semantic (general vs. specific), and directly supports both interpretability and downstream specialization.

Generalist pretraining and fine-tuning. The paradigm of pretraining a generalist model and adapting it to specific downstream tasks has become central to modern machine learning and robotics (Palan et al., 2019). Jaques et al. (2017) introduces KL-control fine-tuning to prevent catastrophic forgetting of a pretrained prior, a technique later adopted for aligning LLMs with human preferences (Ouyang et al., 2022). Finn et al. (2017) learns a parameter initialization for rapid adaptation, and Rakelly et al. (2019) extends this by inferring latent task representations without task labels at test time. In contrast, GRID instantiates this paradigm through reward structure: the generalist is trained on the shared, general reward, and specialization consists of fine-tuning on the specific reward, which encodes individual preferences. This grounds the prior in a semantically meaningful decomposition, making the source of generalization transparent and the adaptation target well-defined.

3 PRELIMINARIES

Markov Decision Process. Given a domain with heterogeneous tasks, the goal is to learn a policy that enacts shared environmental norms from heterogeneous demonstrations of those tasks. We assume N demonstrators act according to their own task p. Each demonstrator provides M demonstrations for a total of $N \times M$ demonstrations within the same domain. For each demonstrator's task, we model their environment as a Markov Decision Process, denoted by $\langle \mathcal{S}, \mathcal{A}, R^{(p)}, T, \gamma \rangle$; $\mathcal{S}, \mathcal{A}$ represents the state and action space respectively; $R^{(p)}(s, a)$ is the reward for demonstrator p for taking action a in state s; $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function that yields the probability of entering the next state s' from the current state s by applying action a; and the total discounted reward over trajectory $\tau = \langle s_0, a_0, ..., s_T, a_T \rangle$ is $R^{(p)}(\tau) = \sum_{t=0}^{T} \gamma^t R^{(p)}(s_t, a_t)$ where $\gamma \in [0, 1)$ is the discount factor. The goal of reinforcement learning is to learn an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected future reward: $\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim \pi}[R(\tau)]$. Ziebart (2010) introduces the principle of maximum entropy that appends an entropy bonus to the standard cost function to incentivize exploration: $\pi^* = \arg\max_\pi \mathbb{E}_{\tau \sim \pi}[\sum_{t=0}^{T} R(s_t, a_t) + \alpha H(\pi(\cdot|s_t))]$.

Inverse Reinforcement Learning. Standard IRL operates within an MDP$\backslash R$ and infers a reward function that explains a set of demonstrated trajectories $\mathcal{D} = \{\tau_1, .., \tau_M\}$. Maximum entropy IRL infers a reward function that exponentially favors the demonstrated trajectories according to the

probability distribution: $p_\theta(\tau) \propto \exp(\sum_{t=0}^{T} \gamma^t R_\theta(s_t, a_t))$. Then, the IRL procedure seeks to maximize the likelihood: $\max_\theta \mathbb{E}_{\tau\sim\mathcal{D}}[\log p_\theta(\tau)]$. In our problem set up, the dataset provided is $\mathcal{D} = \{\mathcal{D}^1, ..., \mathcal{D}^N\}$ where $\mathcal{D}^p = \{\tau_1^p, ...\tau_M^p\} \forall p \in \{1, ..., N\}$. We can perform IRL on each $\mathcal{D}^p$ to yield $R^{(p)}$ for each task.

Information Bottleneck. Given an input X and output Y, the goal of an information bottleneck (IB) is to learn an encoding Z from X that is maximally expressive about Y while being maximally compressive about X (Tishby et al., 1999). The IB principle (1) describes expressiveness and compressibility with mutual information, I, and describes their tradeoff with a positive hyperparameter β. However, IB is difficult to implement in practice because computing the mutual information is challenging. To address this, Alemi et al. (2019) develop a deep variational information bottleneck (VIB) using variational inference to bound the mutual information. These bounds are described by a variational decoder $q(y|z)$, the entropy of Y, $H(Y)$, and the KL divergence between the encoder distribution $p(Z|X)$, and a prior distribution $r(z)$ (see (2) and (3)).

$$\mathcal{L}_{IB} = I(Z;Y) - \beta I(Z;X) \tag{1}$$

$$I(Z;Y) \geq \mathbb{E}_{y,z\sim p(y,z)}[\log q(y|z)] + H(Y) \tag{2}$$

$$I(Z;X) \leq \mathbb{E}_{x\sim p(x)}[D_{KL}(p(z|x)||r(z))] \tag{3}$$

4 General Reward Inference and Disentanglement

We present General Reward Inference and Disentanglement (GRID), a novel IRL framework (Fig. 1) and its architecture (Fig. 2) that can be used to train a generalist agent by extracting the general reward signal from a heterogeneous population of demonstrators performing different tasks in the environment. GRID uses an information-theoretic technique called the variational information bottleneck to disentangle $R^{(p)}$ into R_g and $R_s^{(p)}$.

4.1 Problem Setup

We consider multiple tasks with different reward functions in the same domain. We assume that provided demonstrations are optimal with respect to the individual tasks and representative of environmental norms. Behaviors that are not present in these demonstrations are then not representative of environmental norms. After we acquire a reward signal $R^{(p)}$ for each demonstrated task p, we seek to disentangle the general reward R_g from the specific reward $R_s^{(p)}$. We call the shared component the general reward because it describes how any agent should *generally* act in the domain, and we call the individual component the *specific* reward because it describes the agent's specific task. Thus, the reward corresponding to task p is $R^{(p)} = R_g + R_s^{(p)}$. By definition, the specific $R_s^{(p)}$ is defined by task p, meaning the inputs are the task identifier and the states and actions. Additionally, we seek to learn the rewards such that $R_s^{(p)}$ contains the minimum amount of shared information across all tasks. In other words, $R_s^{(p)}$ should be sparse so R_g can encompass all of the domain information relevant to environmental norms and general capabilities. R_g is completely independent of what task is performed and is purely a function of states and actions. We parameterize $R_s^{(p)}$ and R_g as neural network parameters θ_s and θ_g. Lastly, we embed each task identity p into a learnable embedding vector $\omega^{(p)} \in \mathbb{R}^d$ for greater expressivity. The overall function decomposition becomes:

$$R^{(p)} = R_g(s, a; \theta_g) + R_s^{(p)}(\omega^{(p)}, s, a; \theta_s) \tag{4}$$

4.2 GRID Architecture and Target Requirements

Following (4), the GRID architecture (Fig. 2) features two neural network blocks parameterized by θ_g and θ_s, that represent R_g and $R_s^{(p)}$ respectively, whose outputs are summed to reconstruct the total reward: $\hat{R}^{(p)}(\omega^{(p)}, s, a) = R_g(s, a) + R_s^{(p)}(\omega^{(p)}, s, a)$. The architecture is trained to satisfy three requirements: **(R1)** The specific reward $R_s^{(p)}(s, a, \omega^{(p)})$ captures behaviors relevant to corresponding agent p. **(R2)** The general reward captures behaviors that are meaningful to *all*

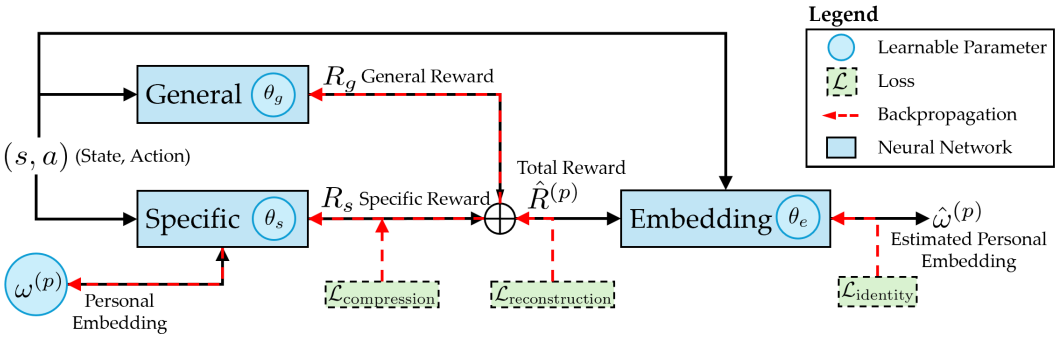


Figure 2: The model architecture of GRID. The summed outputs of the general and specific reward models yield the total reward. $\mathcal{L}_{\text{compression}}$ and $\mathcal{L}_{\text{identity}}$ form the information bottleneck while $\mathcal{L}_{\text{reconstruction}}$ helps predict the total reward. Losses are jointly optimized.

agents. In other words, the general reward should primarily explain the corresponding behavior of each task, leaving the specific reward to be emphasized only when necessary. **(R3)** It must accurately reproduce the original reward $R^{(p)}$. That is, $R^{(p)} \approx \hat{R}^{(p)}$. We address each of these requirements by jointly optimizing the information bottleneck and reconstruction loss described below.

Information Bottleneck. We utilize the IB formulation to address **(R1)** and **(R2)**. The specific reward should be maximally expressive of the task at hand, p, while maximally compressive so that R_g fully represents environmental norms. Succinctly, we let the input $X = (\omega^{(p)}, s, a)$ be the embedding and the state and action; the encoding $Z = (R_s^{(p)})$ be the specific reward; and the target $Y = (\omega^{(p)})$ be the identifier. While standard VIB defines the decoder purely on the bottleneck $q(Y|Z)$, a scalar reward value of $R_s^{(p)}$ cannot reconstruct a continuous task embedding without environment context. Therefore, we implement a *conditional* formulation of the lower bound, where the auxiliary embedding network q is conditioned on states and actions. Below outlines the resulting bounds, where $q(\omega^{(p)}|R_s^{(p)}, s, a; \theta_e)$ is the auxiliary embedding network parameterized by θ_e and $r(z)$ is a chosen prior representative of the specific reward.

$$I(R_s; \omega^{(p)}|s, a) \geq \mathbb{E}_{s,a,p\sim D}[\log q(\omega^{(p)}|R_s^{(p)}, s, a)] + H(\omega^{(p)} \mid s, a) \tag{5}$$

$$I(R_s; \omega^{(p)}, s, a) \leq \mathbb{E}_{s,a,p\sim D}[D_{KL}(p(R_s^{(p)}|\omega^{(p)}, s, a)||r(z))] \tag{6}$$

Maximizing (5) is equivalent to minimizing (7) below and backpropagating the gradients into the original $\omega^{(p)}$ (Schrum et al., 2021; 2024). To do so, we sample $\omega^{(p)} \sim \mathcal{N}(0, I_d)$ before training. Then, during training, q attempts to reproduce $\omega^{(p)}$, incentivizing its prediction $\hat{\omega}^{(p)}$ and $\omega^{(p)}$ to converge to each other's values.

$$\mathcal{L}_{\text{identity}}(\theta_s, \theta_q, \omega^{(p)}) = \mathbb{E}_{s,a,p\sim D}\left[\frac{1}{d}\sum_{i=1}^{d}(\hat{\omega}_i^{(p)} - \omega_i^{(p)})^2\right] \tag{7}$$

Choosing $r(z)$ to be the zero-centered Laplace distribution simplifies minimizing (6) into minimizing the $L1$ loss of $R_s^{(p)}$. Alternatively, choosing $r(z)$ to be the zero-meaned Gaussian turns it into minimizing the $L2$ loss of $R_s^{(p)}$. We present the $L1$ variant below.

$$\mathcal{L}_{\text{compression}}(\theta_s) = \mathbb{E}_{s,a,p\sim D}[|R_s^{(p)}(\omega^{(p)}, s, a)|] \tag{8}$$

In summary, the VIB loss ensures the general reward captures as much information as possible, but anything that is not shared among agents is compressed into the agent-specific rewards.

Reconstruction Loss. To address **(R3)**, we minimize the following squared error loss between the predicted total reward $\hat{R}^{(p)}$ and the provided reward $R^{(p)}$ for faithful decomposition of the general and specific components to the observed reward signal.

$$\mathcal{L}_{\text{reconstruction}}(\theta_g, \theta_s, \theta_q, \omega^{(p)}) = \mathbb{E}_{s,a,p\sim D}[(\hat{R}^{(p)}(\omega^{(p)}, s, a) - R^{(p)})^2] \tag{9}$$

Altogether, we combine the IB losses and reconstruction loss to jointly optimize the network parameters: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{reconstruction}} + \lambda_1\mathcal{L}_{\text{identity}} + \lambda_2\mathcal{L}_{\text{compression}}$, where $\lambda_1, \lambda_2 > 0$.

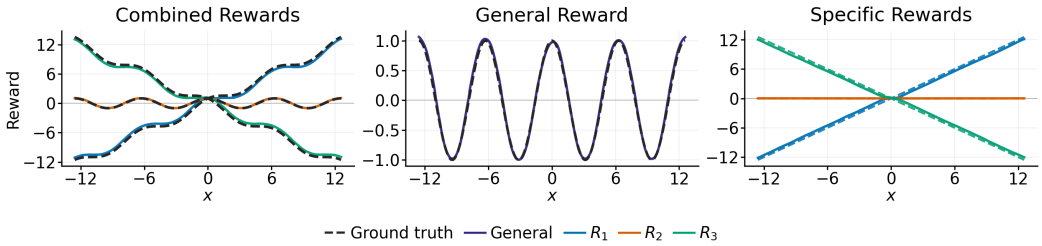


Figure 3: Basis Function Decomposition. Left shows the total reward function. Middle shows the general reward learned by GRID overlapping the ground-truth $\cos(x)$ reward. Right shows the specific rewards learned by GRID overlapping the ground-truth linear $\{+1x, +0x, -1x\}$ rewards. As shown, GRID can automatically and precisely disentangle the sinusoidal and linear behaviors.

4.3 Generalist and Specialist Training

Generalist Training. We train a generalist agent by applying any RL algorithm to the learned general reward component R_g. Following the principle of maximum entropy, RL agents explore the environment during training, potentially leading the agent to states that are relatively out of distribution (OOD) compared to the demonstrations. In this region, GRID's general reward may not extrapolate well, so we develop a mixture of rewards for explored states in the RL process. The mixture contains a $[-1, 1]$ scaled general reward $\tilde{r}(s, a)$ and OOD reward r_{ood} that are weighted by $w(s)$ and $1-w(s)$: $r(s, a) = w(s)\tilde{r}(s, a)+(1-w(s))r_{ood}$. For discrete state space domains, $w(s) = 1$ for encountered states in the demonstrations and $w(s) = 0, r_{ood} = -1$ for never encountered states. For continuous state spaces, we use a k-nearest-neighbors estimate of demonstrated state support. $w(s)$ is close to 1 for states near demonstrated states and approaches 0 as the state moves away from demonstrated states. We set $r_{ood} \leq -1$, so the agent moves toward in-distribution states that are accurately characterized by GRID. This is similar to Reddy et al. (2020)'s idea: states in the demonstrations are rewarded +1 while all other states are rewarded 0.

Specialist Training. Once the generalist is trained, we fine-tune the generalist policy to create a specialist agent. We use Jaques et al. (2017)'s KL-control method to fine-tune the generalist policy π_g for a new task defined by a new specialist reward function $R_{FT}(s_t, a_t)$. The fine-tuning objective maximizes the reward while penalizing divergence from the generalist prior π_g: $\mathcal{L}(\pi_{FT}) = \mathbb{E}_\pi \left[\sum_t R_{FT}(s_t, a_t) + \beta\big(\log \pi_g(a_t \mid s_t) - \log \pi_{FT}(a_t \mid s_t)\big)\right]$. The KL penalty prevents catastrophic forgetting of the general competencies encoded in π_g, while allowing the specialist policy to deviate toward agent-specific preferences. This mirrors the same use of fine-tuning in language models (Ouyang et al., 2022), but grounds the prior in a semantically decomposed reward signal rather than a pretraining corpus.

5 Experiments

5.1 Domains

Basis Function Decomposition. A simple synthetic continuous domain with known ground truth designed to provide clear intuition for reward disentanglement (see Fig. 3). We define three total reward functions: $R_1(x) = \cos(x) + x$, $R_2(x) = \cos(x)$, and $R_3(x) = \cos(x) - x$ where the general reward is $\cos(x)$ and specific rewards are $\{+1x, +0x, -1x\}$. GRID is trained to reconstruct these rewards over the range $[-4\pi, 4\pi]$. We additionally inject noise into ground truth values to assess robustness. We report the mean squared error (MSE) for the general reward.

Craftax. An 8×8 multi-agent discrete world, representing a simplified version of the Minecraft game (Ye & Jaques, 2024). Three agents each pursue a unique objective: collect wood, stone, or diamond, and all share a penalty for stepping on lava. This domain tests GRID's ability to disentangle a shared safety constraint from different individual goals in a discrete state-action space. We visualize the reward distribution across the state space for general and specific rewards.

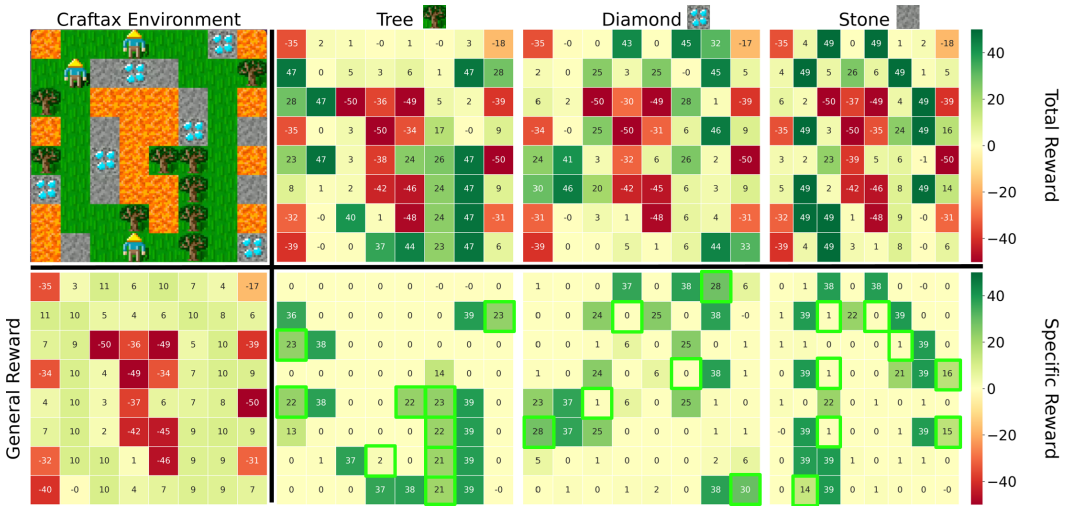


Figure 4: Craftax Domain. We display $\max_a R(s, a)$ for total, specific, and general rewards learned by GRID over the state space for each agent. Penalties are issued for being on the lava tile while collecting the specified item (tree, diamond, or stone) is rewarded. Items can be collected by either being on the actual tile or its adjacent tiles. Tiles with the specific item are boxed in neon green for visual aid. The general reward incentivizes avoiding tiles with lava, while the specific rewards incentivize mining for trees, diamonds and stone on tiles close to those resources, showing GRID effectively learns shared safety norms while preserving individual tasks.

Highway-Env. A continuous state and discrete action space autonomous driving simulator, modified into a three-agent setting with four lanes and additional moving vehicles (Leurent & contributors, 2018). All agents share a general reward for maintaining high speed and avoiding crashes. Each agent additionally holds an easily definable preference: Agent i prefers Lane i, for $i = 0, 1, 2$. Lane 3 is deliberately withheld from training to evaluate downstream specialization to an unseen preference. We perform AIRL for each agent to acquire rewards.

MuJoCo. Two continuous state and action domains: pendulum and hopper (de Lazcano et al., 2024). Traditionally, these domains have a single reward for a single task. To create multiple diverse tasks, we follow Chen et al. (2020a)'s example of linearly combining the main environment reward with the objective from Diversity is All You Need (DIAYN) (Eysenbach et al., 2019) which learns a discriminator $q_\phi(z|s)$ that distinguishes various behaviors according to the agent's state as part of the reward, thereby promoting unique behaviors.

$$r_z(s, a) = \log q_\phi(z|s) - \log p(z) \tag{10}$$

To develop broader heterogeneous behavior, we heavily emphasize the DIAYN reward when combined with the main environment reward. Overall, we generate 19 tasks for pendulum and 13 tasks for hopper. We perform AIRL on each task to acquire rewards.

5.2 Baselines and Comparison Metrics.

For comparison methods, we consider standard LfD approaches, behavior cloning (BC) (Torabi et al., 2018) and Adversarial Inverse Reinforcement Learning (AIRL) (Fu et al., 2018), which aim to learn a policy from the heterogeneous demonstrations. We also compare against MSRD (Chen et al., 2020a), FLAIR[2] (Chen et al., 2022), and REPRESENT (Xue et al., 2025) which are most similar to GRID (see Sec. 2) as they too learn a decomposed reward from heterogeneous demonstrations but of the *same* task. For the highway-env and MuJoCo environments, we compare episode returns as a general notion of success. For highway-env specifically, we report lane metrics: lane entropy and lane 3 steps. Entropy measures the generalist's bias. Having a strong bias toward certain states (low entropy) is an indicative heuristic of having more difficulty fine tuning to a new preference (lane 3).

[2]For FLAIR specifically, we do not learn policies sequentially but in parallel since GRID is not a life long learning algorithm.

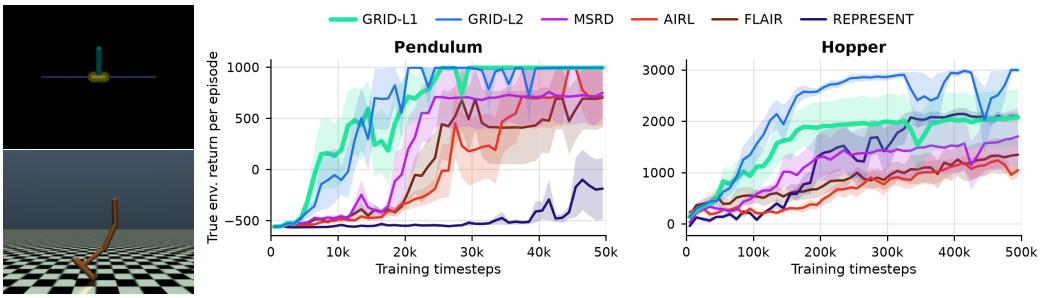


Figure 5: Left figure depicts the pendulum and hopper domains. Middle and right plots show SAC returns throughout training for both respectively. SAC is trained on GRID's general reward, MSRD's task reward and AIRL's learned reward. GRID-L2 variant achieves the best performance.

We evaluate specialist capabilities by training the generalist policy with KL control. In highway-env, R_{FT} is the lane 3 reward while in MuJoCo, R_{FT} is the DIAYN reward. The total of the true domain rewards and fine tune rewards are used to evaluate the specialist policy.

Table 1: Generalist and specialist comparisons in the Highway-Env. See Tables 4 and 5 in the Appendix for the full lane distributions. Means and standard errors are shown.

	Generalist		Specialist			
Method	**General Reward (↑)**	**Entropy (↑)**	**General Reward (↑)**	**Specialist Reward (↑)**	**Total Reward (↑)**	**Lane 3 Steps (↑)**
BC (Torabi et al., 2018)	7.12 ± 0.24	$\mathbf{1.84 \pm 0.01}$	3.94 ± 0.29	$\mathbf{3.44 \pm 0.06}$	7.39 ± 0.26	$\mathbf{34.5 \pm 0.6}$
AIRL (Fu et al., 2018)	12.62 ± 0.43	1.62 ± 0.09	6.92 ± 0.46	3.30 ± 0.06	10.23 ± 0.40	33.1 ± 0.6
MSRD (Chen et al., 2020a)	12.43 ± 0.28	1.70 ± 0.11	6.56 ± 0.29	3.36 ± 0.09	9.91 ± 0.23	33.7 ± 0.9
FLAIR (Chen et al., 2022)	12.38 ± 0.07	1.82 ± 0.06	7.12 ± 0.34	3.26 ± 0.11	10.37 ± 0.24	31.4 ± 0.7
REPRESENT (Xue et al., 2025)	12.38 ± 0.29	1.77 ± 0.04	8.00 ± 0.91	3.05 ± 0.13	11.05 ± 0.89	30.6 ± 1.3
GRID-L1 (ours)	$\mathbf{13.41 \pm 0.18}$	1.77 ± 0.04	8.40 ± 0.23	2.97 ± 0.10	11.37 ± 0.24	29.8 ± 1.0
GRID-L2 (ours)	12.68 ± 0.36	1.74 ± 0.07	$\mathbf{8.60 \pm 0.53}$	2.83 ± 0.03	$\mathbf{11.43 \pm 0.51}$	28.5 ± 0.3

6 Results and Discussion

Basis Function Decomposition. The learned general and specific functions using GRID are shown in Fig. 3. We can see that the disentangled $\cos(x)$ behavior in the general reward and the linear behaviors in the specific rewards are well-recovered by GRID. We note that while GRID performs supervised learning using the ground truth total reward signal, the general and specific rewards are learned in an unsupervised manner and that the $L1$ variant performs better than $L2$ in this domain. With no noise injection on the ground truth, the MSE of the general reward is 0.0018. Even with a $\mathcal{N}(0, 1)$ Gaussian noise injection, the corresponding error is 0.0298. Additional sets of basis functions are in the App. B.

Craftax. We show clear agent behavior disentanglement in Fig. 4. Since the gridworld has discrete states and actions, we report $\max_a R(s, a)$ in each grid cell. The general reward correctly identifies lava locations as shared negative penalties across all agents, while item-gathering preferences (tree wood, diamond, and stone) are cleanly partitioned into the agent-specific reward maps with no observable overlap. We note that the specific reward does not necessarily coincide exactly with the exact item locations, because items can be collected (i.e., reward gained) if the agent is adjacent to those items. Thus, the maximum reward for any tile adjacent to an item can be gained by taking the collect item action. In contrast, the agent takes a penalty regardless of the action taken if it is on lava.

Highway-Env. We evaluate the entire framework on this domain with interpretable rewards to demonstrate GRID's effectiveness. Table 1 shows the full results. We collect demonstrations from agents $0 - 2$, perform AIRL, apply GRID to learned rewards, and train generalist and specialist (prefer lane 3) policies using DQN (Mnih et al., 2013). While the BC prior has the highest lane entropy and occupies lane 3 the most in the fine-tuning stage, it offers the worst overall performance.

Table 2: Generalist and fine-tuning performance on MuJoCo environment. We fine tune generalist priors to new tasks using SAC and KL-control. We fine tune one new task in pendulum for 30k steps and two new tasks in hopper for 100k steps. The best seed-averaged returns across training and peak returns are reported across 5 seeds. GRID tends to perform similarly if not better than baselines.

Method	Pendulum General@50k Steps	Hopper General@500k Steps	Pendulum New Task		Hopper New Task 1		Hopper New Task 2	
			Seed-Avg	Peak	Seed-Avg	Peak	Seed-Avg	Peak
BC (Torabi et al., 2018)	–	–	1399.0 ± 14.9	1427.9	590.6 ± 167.6	936.3	1433.6 ± 676.8	3062.3
AIRL (Fu et al., 2018)	713.2 ± 278.8	1000.8 ± 125.8	1402.7 ± 11.0	1425.6	748.1 ± 81.1	964.1	266.3 ± 306.2	1526.6
MSRD (Chen et al., 2020a)	731.6 ± 258.7	1682.1 ± 551.5	1398.4 ± 15.6	**1452.2**	646.1 ± 175.4	987.3	1547.5 ± 803.7	3709.3
FLAIR (Chen et al., 2022)	696.1 ± 295.3	1338.2 ± 312.0	1399.5 ± 9.8	1442.8	**801.2 ± 19.9**	866.7	1321.9 ± 524.4	3014.3
REPRESENT (Xue et al., 2025)	-192.5 ± 298.7	2099.1 ± 71.2	1372.1 ± 11.9	1447.4	766.8 ± 15.0	812.1	1821.4 ± 517.7	3101.2
GRID-L1 (ours)	993.6 ± 0.9	2072.1 ± 548.7	**1405.5 ± 14.2**	1440.5	680.4 ± 355.5	**2088.9**	2006.3 ± 624.8	3788.8
GRID-L2 (ours)	**994.1 ± 0.5**	**2997.6 ± 40.9**	1402.9 ± 16.9	1449.6	678.1 ± 113.8	1001.2	**2959.8 ± 518.6**	**5228.3**

The BC policies learn to stay in lane 3 but slow down when they are behind another vehicle and do not move around it to maintain high speeds. In contrast, while GRID spends the least amount of steps in lane 3 ($\approx$ 30 of 40 steps), the GRID specialist is still able to maneuver around cars in front of it. In fact, the agent spends less time in lane 3 because it moves around slower vehicles in that lane. AIRL and MSRD sacrifice their agility around other vehicles to stay in lane 3, resulting in lower general rewards after fine-tuning. FLAIR and REPRESENT outperform those two methods but not GRID.

MuJoCo. We evaluate the GRID framework on significantly more tasks. We use AIRL to learn the DIAYN tasks (19 pendulum and 13 hopper), apply GRID to learned rewards, and train the generalists and specialist policies using SAC (Haarnoja et al., 2018). Here, the rewards are less interpretable, so we only report rewards. Figure 5 displays the environment ground truth learning curves for GRID, MSRD, and AIRL. AIRL has no ability to disentangle a shared reward from a specific reward, so its reward function is forced to explain multiple heterogeneous demonstrations. Thus, it expectantly performs subpar to GRID and MSRD. Both GRID $L1$ and $L2$ variants surpass MSRD's learning ability. We attribute this to GRID's separate staged process of acquiring rewards first and then disentangling rewards, where there is no competition between reward learning and disentangling. In contrast, MSRD, FLAIR and REPRESENT, which are built on multiple AIRL models, have to jointly adversarially learn and distill reward information. We perform one fine tuning task in pendulum and two tasks in hopper. Similar to our previous results, GRID generally performs on par with or better than all baselines as shown in Tab. 2. We note in Hopper New Task 2, GRID-L2's best seed average return is around double that of MSRD and FLAIR, demonstrating GRID's ability to learn a strong generalist prior for fine tuning.

7 Conclusion, Limitations, and Future Work

We introduce General Reward Inference and Disentanglement (GRID), a method for social learning that determines which behaviors are worth acquiring from a heterogeneous population of agents. GRID successfully disentangles reward signals into a general component that captures useful universal behaviors and a specific component that encodes individual preferences. We demonstrate that performing RL on the extracted general reward yields a generalist agent that internalizes fundamental environmental norms and that the generalist serves as a prior prime for fine-tuning. With the exception of highway-env, the domains tested feature agents that typically act independently from one another. Studying more interactive multi-agent settings is an important direction for future work, although learning rewards in these environments may be more challenging. Since GRID requires task rewards, its disentanglement ability is directly related to the quality of the rewards. In this work, GRID receives rewards from AIRL and therefore inherits AIRL's assumption that demonstrations are optimal. Extending GRID to accommodate suboptimal demonstrations, for example, by drawing on the approaches of Chen et al. (2020b) and Xue et al. (2025), would be an interesting direction for future research. More broadly, in the context of social learning, enabling a generalist agent to learn from surrounding agents that are interactive, heterogeneous, and potentially suboptimal presents an exciting opportunity. Additionally, this paper focuses on the general reward and its applications, leaving the specific rewards and task embeddings largely unexplored. Developing methods to use these embeddings to characterize demonstrators could enable further personalization, such as in assistive-AI settings, as well as targeted fine-tuning.

AI USE STATEMENT

In this work, we used generative AI tools to help run RL algorithms from stable-baselines3 to collect data, implement baselines, provide implementation suggestions for handling OOD states, and aid in the IB loss derivation. We have not used generative AI tools for the ideation and implementation of our GRID framework and architecture, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results and formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, and assisting with translation are not applicable to this work.

Additionally, we used generative AI tools for formatting tables and figures, editing writing, and finding related works. We have reviewed all AI-assisted work. For example, RL results were confirmed, visualized, and cross referenced with results from other works and two authors went over the loss derivation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work does not raise ethical concerns.

REPRODUCIBILITY STATEMENT

The implementation of the GRID architecture and training scripts are provided for the basis function decomposition domain.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Int. Conf. on Machine Learning*, 2004.

M. Abdulhai, N. Jaques, and S. Levine. Basis for Intentions: Efficient Inverse Reinforcement Learning using Past Experience. *Available at* `https://arxiv.org/abs/2208.04919`, 2023.

A. Alemi, I. Fischer, J. Dillon, and K. Murphy. Deep variational information bottleneck. In *Int. Conf. on Learning Representations*, 2019.

J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. RUDDER: Return Decomposition for Delayed Rewards. In *Conf. on Neural Information Processing Systems*, 2019.

L. Chen, R. Paleja, M. Ghuy, and M. Gombolay. Joint goal and strategy inference across heterogeneous demonstrators via reward network distillation. In *Proc. ACM/IEEE Int. Conference on Human Robot Interaction*, 2020a.

L. Chen, R. Paleja, and M. Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conf. on Robot Learning*, 2020b.

L. Chen, S. Jayanthi, R. Paleja, D. Martin, V. Zakharov, and M. Gombolay. Fast lifelong adaptive inverse reinforcement learning from demonstrations. In *Conf. on Robot Learning*, 2022.

Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2024. URL http://github.com/Farama-Foundation/Gymnasium-Robotics.

B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *Int. Conf. on Learning Representations*, 2019.

A. Filos, C. Lyle, Y. Gal, S. Levine, N. Jaques, and G. Farquhar. PsiPhi-Learning: Reinforcement Learning with Demonstrations using Successor Features and Inverse Temporal Difference Learning. In *Int. Conf. on Machine Learning*, 2021.

C. Finn, P. Abbeel, and S. Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Int. Conf. on Machine Learning*, 2017.

J. Fu, K. Luo, and S. Levine. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning. In *Int. Conf. on Learning Representations*, 2018.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Int. Conf. on Machine Learning*, 2018.

J. Ho and S. Ermon. Generative Adversarial Imitation Learning. In *Conf. on Neural Information Processing Systems*, 2016.

B.* Ivanovic, K.* Leung, E. Schmerling, and M. Pavone. Multimodal Deep Generative Models for Trajectory Prediction: A Conditional Variational Autoencoder Approach. *IEEE Robotics and Automation Letters*, 6(2):295–302, 2021.

N. Jaques, S. Gu, D. Bahdanau, J. M. Hernández-Lobato, R. E. Turner, and D. Eck. Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control. In *Int. Conf. on Machine Learning*, 2017.

Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez. Explainable reinforcement learning via reward decomposition. In *Int. Joint Conf. on Artificial Intelligence*, 2019.

E. Leurent and contributors. An Environment for Autonomous Driving Decision-Making, 2018. Available at https://github.com/eleurent/highway-env.

Yunzhu Li, Jiaming Song, and Stefano Ermon. InfoGAIL: Interpretable imitation learning from visual demonstrations. In *Conf. on Neural Information Processing Systems*, 2017.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *Conf. on Neural Information Processing Systems*, 2013.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. L.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 Lowe R. Training language models to follow instructions with human feedback. In *Conf. on Neural Information Processing Systems*, 2022.

M. Palan, G. Shevchuk, N. Landolfi, and D. Sadigh. Learning reward functions by integrating human demonstrations and preferences. In *Robotics: Science and Systems*, 06 2019.

S. Poddar, Y. Wan, H. Ivison, A. Gupta, and N. Jaques. Personalizing Reinforcement Learning from Human Feedback with Variational Preference Learning. In *Conf. on Neural Information Processing Systems*, 2024.

A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

K. Rakelly, A. Zhou, D. Quillen, C. Finn, and S. Levine. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. In *Int. Conf. on Machine Learning*, 2019.

D. Ramachandran and E. Amir. Bayesian Inverse Reinforcement Learning. In *Int. Joint Conf. on Artificial Intelligence*, 2007.

S. Reddy, A. Dragan, and S. Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. In *Int. Conf. on Learning Representations*, 2020.

I. Remy, C. Chang, and K. Leung. Learning Probabilistic Responsibility Allocations for Multi-Agent Interactions. *Available at* `https://arxiv.org/abs/2604.13128`, 2026.

S. Ross, G. J. Gordon, and A. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *AI & Statistics*, 2011.

E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction. In *Proc. IEEE Conf. on Robotics and Automation*, 2018.

M. L. Schrum, E. Hedlund, and M. C. Gombolay. Improving Robot-Centric Learning from Demonstration via Personalized Embeddings. In *AAAI Fall Symposium*, 2021.

M. L. Schrum, E. Sumner, M. C. Gombolay, and A. Best. Maveric: A data-driven approach to personalized autonomous driving. *IEEE Transactions on Robotics*, 40:1952–1965, 2024.

E. Seraj, KM. Lee, Z. Zaidi, Q. Xiao, Z. Li, A. Nascimento, S. Van Waveren, P. Tambwekar, R. Paleja, D. Das, and M.C. Gombolay. Interactive and Explainable Robot Learning: A Comprehensive Review. *Foundations and Trends in Robotics*, 12(2–3):75–349, 2024.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Allerton Conference on Communication, Control and Computing*, 1999.

F. Torabi, G. Warnell, and P. Stone. Behavioral Cloning from Observation. In *Int. Joint Conf. on Artificial Intelligence*, 2018.

H. van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang. Hybrid Reward Architecture for Reinforcement Learning. In *Conf. on Neural Information Processing Systems*, 2017.

C. Xue, L. Chen, and M. Gombolay. Better than diverse demonstrators: Reward decomposition from suboptimal and heterogeneous demonstrations. In *IEEE Robotics and Automation Letters*, 2025.

E. Ye and N. Jaques. An Efficient Open World Benchmark for Multi-Agent Reinforcement Learning. In *Conf. on Neural Information Processing Systems - Workshop on Open-World Agents*, 2024.

L. Yu, T. Yu, C. Finn, and S. Ermon. Meta-Inverse Reinforcement Learning with Probabilistic Context Variables. In *Conf. on Neural Information Processing Systems*, 2019.

B. D. Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, 2010.

B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI Conf. on Artificial Intelligence*, 2008.

A Derivation of Loss Functions

We aim to maximize the information bottleneck objective (Tishby et al., 1999).

$$I(Z,Y) - \beta I(Z,X) \tag{11}$$

Following Alemi et al. (2019)'s methodology and notation of information bottlenecks, we assume that the joint distribution of X, Y, Z factors into the following:

$$p(X,Y,Z) = p(Z|X,Y)p(Y|X)p(X) = p(Z|X)p(Y|X)p(X) \tag{12}$$

and that $p(Z|X,Y) = p(Z|X)$.

The mutual information between Y and Z is

$$I(Z;Y) = \int dydz\, p(y,z) \log \frac{p(y,z)}{p(y)p(z)} = \int dydz\ \log \frac{p(y|z)}{p(y)} = H(Y) - H(Y|Z) \tag{13}$$

where $p(y|z)$ is defined as

$$p(y|z) = \int dx p(x, y|z) = \int dx\, p(y|x)p(x|z) = \int dx\, \frac{p(y|x)p(z|x)p(x)}{p(z)}. \tag{14}$$

However, recovering $p(y|z)$ is intractable so we let $q(y|z)$ be its variational approximation. Since KL divergence is always nonnegative, $D_{KL}(p(Y|Z)||q(Y|Z)) \geq 0$, the following inequality holds:

$$\int dy\, p(y|z) \log p(y|z) \geq \int dy\, p(y|z) \log q(y|z) \tag{15}$$

Thus, $I(Z;Y)$ is bounded from below:

$$\begin{aligned} I(Z;Y) &\geq \int dydz\, p(y,z) \log \frac{q(y|z)}{p(y)} \\ &= \int dydz\, p(y,z) \log q(y|z) - \int dy\, p(y) \log p(y) \\ &= \int dydz\, p(y,z) \log q(y|z) + H(Y) \\ &\approx \mathbb{E}_{y\sim Y, z\sim Z} \log q(y|z) + H(Y) \end{aligned} \tag{16}$$

Maximizing the mutual information between Z and Y is then equivalent to maximizing $\mathbb{E}_{y\sim Y,z\sim Z} \log q(y|z)$ since $H(Y)$ is constant. We let $q(y|z)$ take the form of a Multivariate Gaussian distribution with fixed variance σ^2 and assume the network θ_q that parametrizes this distribution outputs the mean $\hat{\omega}$. Following (Schrum et al., 2021; 2024), this transforms the mutual information loss into a squared error objective:

$$q(\omega^{(p)}|s, a, R_s) = \frac{1}{(\sqrt{2\pi\sigma^2})^d} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{d} (\hat{\omega}_i^{(p)} - \omega_i^{(p)})^2 \right) \tag{17}$$

We log both sides to derive the following equation.

$$\begin{aligned} \log q(\omega^{(p)}|s, a, R_s) &= \log \frac{1}{(\sqrt{2\pi\sigma^2})^d} + \log \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{d} (\hat{\omega}_i^{(p)} - \omega_i^{(p)})^2 \right) \\ \log q(\omega^{(p)}|s, a, R_s) &= \log \frac{1}{(\sqrt{2\pi\sigma^2})^d} + \left(-\frac{1}{2\sigma^2} \sum_{i=1}^{d} (\hat{\omega}_i^{(p)} - \omega_i^{(p)})^2 \right) \end{aligned} \tag{18}$$

Maximizing the previous equation (18) renders out the first term and leaves the squared error term to be minimized. Scaling by $1/d$ leads to (7).

$$\begin{aligned} \arg\max_{\theta_q} \log q(\omega|s, a, R_s) &= \arg\max_{\theta_q} \left\{ -\frac{1}{2\sigma^2} \sum_{i=1}^{d} \hat{\omega}_i^{(p)} - \omega_i^{(p)})^2 \right\} \\ \arg\max_{\theta_s} \log q(\omega|s, a, R_s) &= \arg\min_{\theta_q} \sum_{i=1}^{d} (\hat{\omega}_i - \omega_i)^2 \end{aligned} \tag{19}$$

Again, following Alemi et al. (2019), we now turn our attention toward the compression term. We aim to minimize the mutual information between X and Z defined as

$$\begin{aligned} I(X;Z) &= \int dxdz\, p(x,z) \log \frac{p(z|x)}{p(z)} \\ &= \int dzdx\, p(x,z) - \int dz\, p(z) \log p(z) \\ &\approx \mathbb{E}_{x\sim p(x)}[D_{KL}(p(z|x)||p(z))] \end{aligned} \tag{20}$$

We introduce a variational prior $r(z)$ to approximate $p(z)$, the true intractable marginal distribution of the specific rewards. Since the KL divergence between the true marginal and chose prior $r(z)$ is always non negative, we establish a strict upper bound:

$$\begin{aligned} I(X;Z) &\leq \mathbb{E}_{x\sim p(x)}[D_{KL}(p(z|x)||r(z))] \\ &= -H(Z|X) - \mathbb{E}_{x,z\sim p(z,x)}[\log r(z)]] \end{aligned} \tag{21}$$

Following our assumption that general rewards constitute the majority of behavior and that specific rewards are then naturally more sparse, we assume $r(z)$ takes the form of a zero-centered Laplace prior: $r(z) = \frac{1}{2b}\exp(-\frac{|z|}{b})$. If instead we assume that the specific behaviors follow a normal distribution, then $r(z) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{z^2}{2\sigma^2})$. The entropy term is constant because our neural network is deterministic. Minimizing (21) yields the L1 loss.

$$\begin{aligned}\min_{\theta_s} -\mathbb{E}_{x\sim p(x)}\mathbb{E}_{z\sim p(z|x)}[\log r(z)] &\leq \min_{\theta_s} -\mathbb{E}_{x\sim p(x)}\mathbb{E}_{z\sim p(z|x)}\left[\log\left(\frac{1}{2b}\right) - \frac{|z|}{b}\right] \\ &\leq \min_{\theta_s} \mathbb{E}_{x\sim p(x)}\mathbb{E}_{z\sim p(z|x)}[|z|]\end{aligned} \tag{22}$$

B Additional Basis Function Decomposition Results

We consider the following set of equations:

$$\begin{aligned}R_1(x) &= \cos(x) + 0.5\sin(x) \\ R_2(x) &= \cos(x) \\ R_1(x) &= \cos(x) - 0.5\sin(x)\end{aligned} \tag{23}$$

The MSE between $\cos(x)$ and the general reward is 0.0058.

We consider another set of functions to assess non-additive structures.

$$\begin{aligned}R_1(x) &= -0.5\cos(x) \\ R_2(x) &= \cos(x) \\ R_1(x) &= 0.5\cos(x)\end{aligned} \tag{24}$$

The MSE between $\cos(x)$ and the general reward is 0.1577. Although the MSE is larger, the recovered reward is still sinusoidal, achieving an angular frequency of 0.99. The ground truth is 1.

We inject Gaussian noise of increasing variance to assess GRID's robustness on the original set of basis functions: $\{\cos(x) - x, \cos(x), \cos(x) + x\}$.

Table 3: Noise Injection

σ^2	0	0.1	0.5	1	5
$\cos(x)$ MSE	0.0018	0.0027	0.0183	0.0298	0.2228

C Additional Highway Results

Table 4: Generalist Comparisons in the Highway-Env.

Method	**Reward (↑)**	**Episode Length (↑)**	**Lane Entropy (↑)**	**Lane Occupancy**			
				Lane 0	**Lane 1**	**Lane 2**	**Lane 3**
BC	7.12 ± 0.24	21.3 ± 0.6	$\mathbf{1.84 \pm 0.01}$	0.21 ± 0.01	0.36 ± 0.02	0.34 ± 0.02	0.09 ± 0.01
Pooled AIRL	12.62 ± 0.43	34.0 ± 1.0	1.62 ± 0.09	0.34 ± 0.09	0.39 ± 0.03	0.21 ± 0.06	0.06 ± 0.02
MSRD	12.43 ± 0.28	33.1 ± 0.7	1.70 ± 0.11	0.31 ± 0.08	0.39 ± 0.04	0.22 ± 0.03	0.08 ± 0.03
GRID-L1 (ours)	$\mathbf{13.41 \pm 0.18}$	$\mathbf{35.4 \pm 0.4}$	1.77 ± 0.04	0.13 ± 0.03	0.49 ± 0.04	0.26 ± 0.03	0.12 ± 0.03
GRID-L2 (ours)	12.68 ± 0.36	33.6 ± 0.8	1.74 ± 0.07	0.16 ± 0.04	0.48 ± 0.03	0.25 ± 0.03	0.11 ± 0.02

D Experiment Setup

D.1 Basis Function Decomposition Experiment

Data Generation. For the synthetic basis function decomposition experiment, we generate 2000 demonstration samples uniformly distributed across the domain $[-4\pi, 4\pi]$.

Table 5: Specialist Comparisons in the Highway-Env.

Method	Reward (↑)	Episode Length (↑)	Lane 3 Steps (↑)	Lane Occupancy			
				Lane 0	Lane 1	Lane 2	Lane 3
BC	7.39 ± 0.26	36.5 ± 0.5	$\mathbf{34.5 \pm 0.6}$	0.01 ± 0.00	0.02 ± 0.00	0.03 ± 0.00	0.94 ± 0.01
Pooled AIRL	10.23 ± 0.40	$\mathbf{37.3 \pm 0.2}$	33.1 ± 0.6	0.03 ± 0.01	0.03 ± 0.01	0.06 ± 0.01	0.89 ± 0.02
MSRD	9.91 ± 0.23	37.2 ± 0.5	33.7 ± 0.9	0.02 ± 0.01	0.03 ± 0.01	0.05 ± 0.01	0.91 ± 0.01
GRID-L1 (ours)	11.37 ± 0.24	36.2 ± 0.6	29.8 ± 1.0	0.01 ± 0.00	0.05 ± 0.01	0.12 ± 0.01	0.82 ± 0.02
GRID-L2 (ours)	$\mathbf{11.43 \pm 0.51}$	35.0 ± 0.4	28.5 ± 0.3	0.01 ± 0.00	0.05 ± 0.01	0.13 ± 0.01	0.81 ± 0.01

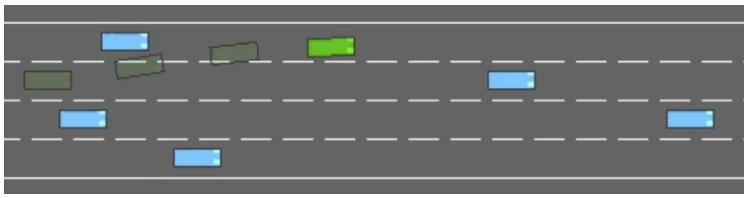


Figure 6: Generalist agent (green vehicle) navigates around other cars (blue vehicles) at high speeds.

D.2 Craftax Experiment

Environment Setup. To aid in visualizing GRID's disentanglement, we constrain the original Craftax environment. The modifications are as follows:

- The environment map is reduced from a 64×64 grid to an 8×8 grid.
- The action space for all agents is restricted to *no-op*, *up*, *down*, *left*, *right*, and *do-action*.
- The original single-step reward formulation is replaced with a cumulative reward structure.
- Adversarial entities (e.g., skeletons, zombies, cows, arrows) are removed.
- Agents no longer require tools to retrieve resources such as stone or diamonds.
- World generation randomization is largely disabled; only agent spawn positions remain randomized.
- Reward weights are modified to ensure each agent prioritizes a distinct resource-gathering task.
- Episodes terminate immediately once an agent completes its specific task or is terminated.

Data Generation. We generate demonstration data using the IPPO implementation from (Ye & Jaques, 2024). To capture a diverse range of skill levels, we collect a total of 2000 demonstration trajectories (500 trajectories each from policy checkpoints at 1, 10, 100, and 200 training iterations). The IPPO actor-critic network utilizes a standard feed-forward architecture comprising three hidden layers of sizes [64, 64, 32] with Tanh activations. The IPPO training hyperparameters are detailed in Table 9.

D.3 Highway-Env Experiment

Environment Setup. We adapt the standard single-agent Highway-Env into a multi-agent environment featuring three agents navigating a four-lane highway. To define the specific behaviors of our heterogeneous population, Agent 0 is rewarded for occupying Lane 0, Agent 1 for Lane 1, and Agent 2 for Lane 2. Lane 3 is left un-preferred by all agents to serve as a holdout specialization target. All lane specialization rewards are valued at 0.1. Each episode runs for at most 40 steps.

Data Generation. We modify the PPO implementation from Stable-Baselines3 (Raffin et al., 2021) to support IPPO for our multi-agent setup. Table 10 lists the hyperparameters used to train the expert demonstrators. Unlisted parameters are default.

Table 6: Network architectures and optimization hyperparameters across methods.

Algorithm	Network Architectures	Optimization & Training Parameters
GRID	$R_g(s,a)$: MLP [1024, 1024, 1024, 1024], ReLU $R_s(\omega^{(p)},s,a)$: MLP [64, 64], ReLU ($\omega^{(p)} \in \mathbb{R}^2$) $q(R_s^{(p)},s,a)$: MLP [128, 128] $\to \hat{\omega}^{(p)} \in \mathbb{R}^2$, ReLU	Adam lr 1×10^{-3}, batch size 128, 100 epochs over demo pairs. Targets scaled to $[-1,1]$.
Per-skill AIRL	Reward $g(s,a)$: MLP [32] Potential $\phi(s)$: MLP [32, 32] Generator policy: MLP [64, 64], tanh	**Disc:** Adam lr 1×10^{-3}, batch 32, 10 steps/round. **Gen:** TRPO (KL 0.001, CG 10, batch 100), 200 steps/round. **Budget:** 500 rounds (MuJoCo), 200 rounds (Highway).
Pooled AIRL	Reward & Potential nets same as above. Generator policy: MLP [64, 64], tanh	**Gen warm start:** BC 2000 steps (lr 3×10^{-4}). **Gen:** TRPO (KL 0.01, batch 100), 1000 steps/round. **Disc:** Adam lr 3×10^{-4}, batch 32, 2 steps/round. 200 rounds.
MSRD	$R_0, \tilde{R}_i$: MLP [32] per net Potential $\phi_0, \tilde{\phi}_i$: MLP [32, 32] per net Generator policies: MLP [64, 64], tanh	**Gen warm start:** BC 6000 steps (lr 3×10^{-4}). **Gen:** TRPO (KL 0.02, batch 256, CG 10), 1000 steps/strat/round. **Disc:** Adam lr 3×10^{-4}, batch 64, 1 step/strat/round. **Rounds:** Pendulum 150, Hopper 100, Highway 200.
FLAIR	Same architectures as MSRD.	Phase 1: Same as MSRD budget. Phase 2: Same as MSRD + BCD step (lr 3×10^{-5}, 1 step/strat/round; Highway: 5 steps/strat/round).
REPRESENT	Phase 1: Same architectures as MSRD. Phase 3 ($R_0, \tilde{R}_i$): MLP [256, 256], ReLU, $\mathbf{W} \sim \mathcal{N}(0, 0.001)$, zero bias.	Phase 1: $\epsilon \in \{0,0,0,0,0,0.05,0.1,0.15,0.2,0.25\}$. Phase 2: $\epsilon \in \{0, 0.05, \ldots, 0.95\}$ (20 levels) $\times$ 5 rollouts. Phase 3: Adam lr 1×10^{-4}, weight decay $0.1 \sum \|\mathbf{W}\|_2^2$, 3000 iterations, batch 64, 5000 snippets/strat.

D.4 MuJoCo Experiments

We use Gymnasium InvertedPendulum-v5 and Hopper-v5 for our continuous control tasks. Both tasks are run for 1000 steps and use the default environment reward as the shared reward. The total reward is

$$r(s,a) = r_{\text{environment}}(s,a) + 0.5 \cdot r_{\text{DIAYN}}(s) \tag{25}$$

Pendulum is trained for 100k steps and hopper is trained for 1M steps. 10 demonstrations are collected for each.

Table 7: Reward processing and shaping formulas applied before feeding to the generalist policies.

Method & Env		Reward Transformation / Blending Formula	Parameters
GRID (L1/L2)	*MuJoCo*	**1. Scaling:** Min-max scaled to $[-1, 1]$ based on demo (s, a) pairs. **2. Blending:** $r' = w \cdot r_{\text{scaled}} + (1 - w) \cdot r_{\text{ood}}$ **3. Weight:** $w = \exp(-\max(d - \epsilon, 0)/\sigma)$ where d is the whitened distance to demo states, and ϵ is the 99th percentile of the demos' own distances.	**Pendulum:** 1-NN distance $r_{\text{ood}} = -10, \sigma = 2$ **Hopper:** 50-NN distance $r_{\text{ood}} = -2, \sigma = 4$
GRID (L1/L2)	*Highway*	Same scaling and blending formula as above. d is computed via 50-NN distance on the 25-dim observation.	$r_{\text{ood}} = -2$ $\sigma = 4$
Baselines	*MuJoCo*	Raw learned reward passed directly to SAC (no scaling).	N/A
Baselines	*Highway*	Min-max mapped to $[0, 1]$ on demo pairs, then clipped. *(Ensures staying alive never pays less than crashing)*	N/A

Table 8: Generalist policy architectures and optimization hyperparameters.

Algorithm & Task		Network Architecture	Optimization & Training Parameters
6.1 SAC	(MuJoCo)	**Algorithm:** SAC (`SBX`) **Actor:** MLP [256, 256], ReLU **Critics (2):** MLP [256, 256], ReLU	**Opt:** lr 3×10^{-4}, batch 256, buffer 10^6. $\tau = 0.005, \gamma = 0.99$. **Routine:** Train freq 1, 1 grad step/env step. Learning starts at 100. Auto entropy tuning. **Env:** 1000-step eps, no fall termination. **Budget:** Pendulum 100k, Hopper 1M steps.
6.2 BC Generalist	(MuJoCo)	**Actor:** Squashed-Gaussian MLP [256, 256] (`SBX`)	**Objective:** Max likelihood on all pooled demo pairs. **Opt:** Adam lr 3×10^{-4}, batch 256. **Budget:** 20k steps.
6.3 DQN	(Highway)	**Algorithm:** DQN (`SB3`) **Q-Network:** MLP [256, 256]	**Opt:** lr 5×10^{-4}, batch 32, buffer 15k, $\gamma = 0.95$. Target update every 50 steps. **Routine:** Train freq 1, 1 grad step/env step. Learning starts at 200. **Exploration:** ϵ-greedy decays $1.0 \rightarrow 0.05$ (first 10%). **Env:** 32 subprocess envs. Random learner car. Ends on crash or 40 steps. Budget: 100k steps.
6.4 BC Generalist	(Highway)	**Q-Network:** MLP [256, 256] (Treated as a 5-way action classifier)	**Objective:** Cross-entropy on all 1200 pooled demo pairs. **Opt:** Adam lr 3×10^{-4}, batch 256. **Budget:** 20k steps.

Table 9: IPPO Training Hyperparameters: Craftax Data Generation

Hyperparameter	**Value**
Training iterations per agent	200
Parallel environments	200
Minibatches	16
Learning rate	2.5×10^{-4}
Entropy coefficient	0.02
Max gradient norm	1.0
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Clip coefficient	0.2
Value function coefficient	0.5
Auxiliary loss coefficient	0.1

Table 10: IPPO Training Hyperparameters: Highway-Env Data Generation

Hyperparameter	**Value**
Learning rate	3×10^{-4}
Steps per rollout (n_steps)	2048
Batch size	256
Epochs	10
Discount factor (γ)	0.95
Entropy coefficient	0.05

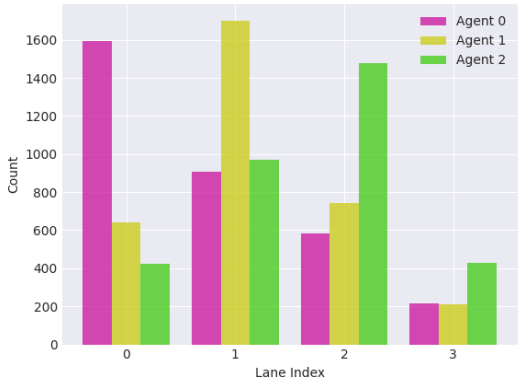


Figure 7: IPPO Agents Lane Counts over 100 Episodes. Average Episode Length=33.0

Table 11: Hyperparameter settings for Soft Actor-Critic (SAC) and the DIAYN Skill Discriminator.

Hyperparameter	**Value**
Soft Actor-Critic (SAC)	
Architecture	MLP [256, 256]
Activation	ReLU
Learning Rate	3×10^{-4}
Replay Buffer Size	10^6
Batch Size	256
Target Smoothing Coefficient (τ)	0.005
Discount Factor (γ)	0.99
Train Frequency	1
Gradient Steps	1 per env step
Skill Discriminator	
Architecture $q(z\|s)$	MLP (width 64, depth 2)
Optimizer	Adam
Learning Rate	3×10^{-4}
Batch Size	256
Replay Buffer Size	10^5
Gradient Steps	1 per env step